## 1.1 Purpose of the Peripheral

The SPI is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (2 to 16 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communication between the device and external peripherals. Typical applications include interface to external I/O or peripheral expansion via devices such as shift registers, display drivers, SPI EPROMS, and analog-to-digital converters.

## 1.2 Terminology Used in This Document

**Table 1-1 Terminology**

| Acronym | Definition |
|---|---|
| DMA | Direct Memory Access |
| LSB | Least significant bit |
| MSB | Most significant bit |
| SPI | Serial peripheral interface |

## 1.3 Features

The SPI has the following features:

- 16-bit shift register
- 16-bit Receive buffer register (SPIBUF) and 16-bit Receive buffer emulation *alias* register (SPIEMU)
- 16-bit Transmit data register (SPIDAT0) and 16-bit Transmit data and format selection register (SPIDAT1)
- 8-bit baud clock generator
- Serial clock (SPICLK) I/O pin
- Slave in, master out (SPISIMO) I/O pin
- Slave out, master in (SPISOMI) I/O pin
- Multiple slave chip select ($\overline{\text{SPISCS[n]}}$) I/O pins (4 pin mode only)
- Programmable SPI clock frequency range
- Programmable character length (2 to 16 bits)
- Programmable clock phase (delay or no delay)
- Programmable clock polarity (high or low)
- Interrupt capability
- DMA support (read/write synchronization events)
- Up to 66 MHz operation

**Note—**Please see the device-specific data manual for the number of slave chip select pins ($\overline{\text{SPISCS[n]}}$) supported.

The SPI allows software to program the following options:

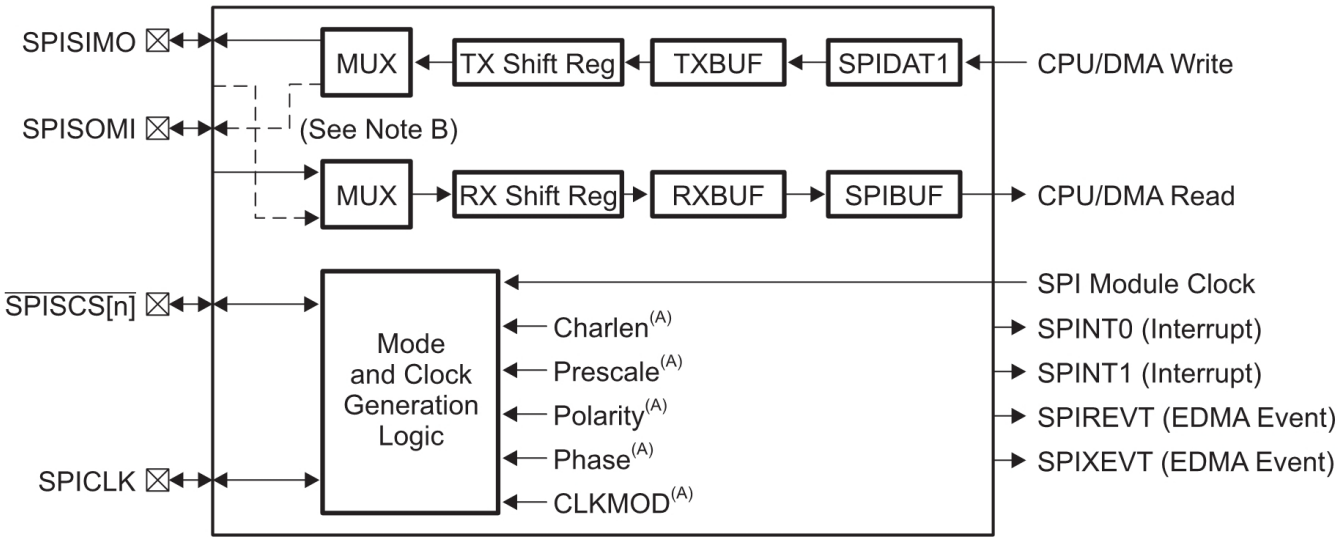
- SPICLK frequency (SPI module clock/2 through SPI module clock/256)
- 3-pin and 4-pin options
- Character length (2 to 16 bits) and shift direction (MSB/LSB first)
- Clock phase (delay or no delay) and polarity (high or low)
- Delay between transmissions in master mode.
- Chip select setup and hold times in master mode
- Chip select hold in master mode

The SPI does not support the following features:

- Multibuffer mode
- Parallel mode or parity
- SPIENA pin
- SPI slave mode
- GPIO mode

## 1.4 Functional Block Diagram

**Figure 1-1 SPI Block Diagram**




**(A)** Indicates the log controlled by SPI register bits.
**(B)** Solid line represents data flow for SPI master mode.

## 1.5 Industry Standard(s) Compliance Statement

The programmable configuration capability of the SPI allows it to gluelessly interface to a variety of SPI format devices. The SPI does not conform to a specific industry standard.

## 2.1 Clock

The SPI clock (SPICLK) is derived from the SPI module clock. The maximum clock bit rate supported is SPI module clock/2, as determined by the PRESCALE field in the SPI data format register n (SPIFMTn). The SPICLK frequency is calculated as:

SPICLK frequency = [SPI module clock] / [SPIFMTn.PRESCALE + 1]

When SPIFMTn.PRESCALE is cleared to 0, the SPICLK frequency defaults to SPI module clock/2.

## 2.2 Signal Descriptions

**Table 2-1 SPI Pins**

| Pin | Type | Function |
|---|---|---|
| SPISIMO | Output | Serial data output in master mode |
| SPISOMI | Input | Serial data input in master mode |
| SPICLK | Output | Serial clock output in master mode |
| $\overline{\text{SPISCS[n]}}$ [1] | Output | Slave chip select output in master mode |

1. The value *n* indicates the SPI pins available; that is, SPISCS[0], SPISCS[1], etc. See the device-specific data manual to determine how many SPI pins are available.

## 2.3 Operation Modes

The SPI operates in master mode. The SPI bus master is the device that drives the SPICLK, SPISIMO, and optionally the $\overline{\text{SPISCS[n]}}$ signals, and therefore initiates SPI bus transfers. The CLKMOD and MASTER bits in the SPI global control register 1 (SPIGCR1) selects master mode. In the master mode, the SPI supports two options:

- 3-pin option
- 4-pin with chip select option

The 3-pin option is the basic clock, data in, and data out SPI interface and uses the SPICLK, SPISIMO, and SPISOMI pins. The 4-pin with chip select option adds the $\overline{\text{SPISCS[n]}}$ pin that is used to support multiple SPI slave devices on a single SPI bus.

## 2.4 Programmable Registers

**Table 2-2 SPI Registers**

| Offset Address [1] | Acronym | Name | Description | Section |
|---|---|---|---|---|
| 0h | SPIGCR0 | Global control register 0 | Contains the software reset bit for the module | Section 3.1 |
| 4h | SPIGCR1 | Global control register 1 | Controls basic configurations of the module | Section 3.2 |
| 8h | SPIINT0 | Interrupt register | Enable bits for interrupts, error, DMA and other functionality. | Section 3.3 |
| Ch | SPILVL | Level register | SPI interrupt levels are set in this register. | Section 3.4 |
| 10h | SPIFLG | Flag register | Shows the status of several events during the operation. | Section 3.5 |
| 14h | SPIPC0 | Pin control register 0 | Determines if pins operate as general I/O or SPI functional pin. | Section 3.6 |
| 38h | SPIDAT0 | Transmit data register 0 | Transmit data register | Section 3.7 |
| 3Ch | SPIDAT1 | Transmit data register 1 | Transmit data with format selection register | Section 3.8 |
| 40h | SPIBUF | Receive buffer register | Holds received word | Section 3.9 |
| 44h | SPIEMU | Receive buffer emulation | Mirror of SPIBUF. Read does not clear flags register. | Section 3.10 |
| 48h | SPIDELAY | Delay register | Sets $\overline{\text{SPISCS[n]}}$ mode, $\overline{\text{SPISCS[n]}}$ pre-transfer / post-transfer delay time. | Section 3.11 |
| 4Ch | SPIDEF | Chip select default register | In $\overline{\text{SPISCS[n]}}$ decoded mode only: sets high low/active $\overline{\text{SPISCS[n]}}$ signal. | Section 3.12 |
| 50h | SPIFMT0 | Format 0 register | Configuration of data word format 0. | Section 3.13 |
| 54h | SPIFMT1 | Format 1 register | Configuration of data word format 1. | Section 3.13 |
| 58h | SPIFMT2 | Format 2 register | Configuration of data word format 2. | Section 3.13 |
| 5Ch | SPIFMT3 | Format 3 register | Configuration of data word format 3. | Section 3.13 |
| 60h | INTVEC0 | Interrupt vector register 0 | Interrupt vector for line INT0. | Section 3.14 |
| 64h | INTVEC1 | Interrupt vector register 1 | Interrupt vector for line INT1. | Section 3.15 |

1. The actual address of these registers is device specific and CPU specific. See the device-specific data manual to verify the SPI register addresses.

## 2.5 Master Mode Settings

The two master mode options are defined by the configuration bit settings listed in Table 2-3. Other configuration bits may take any value in the range listed in Table 2-4. The values listed in Table 2-3 and Table 2-4 should not be changed while the ENABLE bit in the SPI global control register 1 (SPIGCR1) is set to 1. Note that in certain cases the allowed values may still be ignored. For example, Table 2-4 indicates that SPIDELAY may take a range of values in Master 3-pin mode; however, SPIDELAY has no effect in Master 3-pin mode. For complete details on each mode, see the following sections that explain the SPI operation for each of the master modes.

**Table 2-3 SPI Register Settings Defining Master Modes**

| Register | Bit(s) | Master 3-pin | Master 4-pin Chip Select |
|---|---|---|---|
| SPIGCR0 | RESET | 1 | 1 |
| SPIGCR1 | ENABLE | 1 | 1 |
| SPIGCR1 | LOOPBACK | 0 | 0 |
| SPIGCR1 | CLKMOD | 1 | 1 |
| SPIGCR1 | MASTER | 1 | 1 |
| SPIPC0 | SOMIFUN | 1 | 1 |
| SPIPC0 | SIMOFUN | 1 | 1 |
| SPIPC0 | CLKFUN | 1 | 1 |
| SPIPC0 | ENAFUN | 0 | 0 |
| SPIPC0 | SCS0FUN | 0 | 1 |

**Table 2-4 Allowed SPI Register Settings in Master Modes**

| Register | Bit(s) | Master 3-pin | Master 4-pin Chip Select |
|---|---|---|---|
| SPIINT0 | ENABLEHIGHZ | 0,1 | 0,1 |
| SPIFMTn | WDELAY | 0 to 3Fh | 0 to 3Fh |
| SPIFMTn | PARPOL | 0,1 | 0,1 |
| SPIFMTn | PARENA | 0,1 | 0,1 |
| SPIFMTn | WAITENA | 0 | 0 |
| SPIFMTn | SHIFTDIR | 0,1 | 0,1 |
| SPIFMTn | DISCSTIMERS | 0,1 | 0,1 |
| SPIFMTn | POLARITY | 0,1 | 0,1 |
| SPIFMTn | PHASE | 0,1 | 0,1 |
| SPIFMTn | PRESCALE | 0 to FFh | 0 to FFh |
| SPIFMTn | CHARLEN | 2 to 10h | 2 to 10h |
| SPIDELAY | C2TDELAY | 0 to FFh | 0 to FFh |
| SPIDELAY | T2CDELAY | 0 to FFh | 0 to FFh |
| SPIDELAY | T2EDELAY | 0 to FFh | 0 to FFh |
| SPIDELAY | C2EDELAY | 0 to FFh | 0 to FFh |

### 2.5.1 Master Mode Timing Options

The SPI in master mode supports several options to modify the timing of its generation of the chip select signal ($\overline{\text{SPISCS[n]}}$). This allows the SPI to support the timing requirements of various slave devices without adding additional overhead to the CPU by generating the appropriate delays automatically.

#### 2.5.1.1 Chip Select Setup Time

The master can be configured to provide a (slow) slave device a certain chip select setup time to the first edge on SPICLK. This delay is controlled by the C2TDELAY field in the SPI delay register (SPIDELAY) and can be configured between 2 and 257 SPI module clock cycles. The C2TDELAY is applicable only in 4-pin with chip select mode. The C2TDELAY begins when the SPI master asserts $\overline{\text{SPISCS[n]}}$. The C2T delay period is specified by:

Maximum duration of C2TDELAY period = SPIDELAY.C2TDELAY + 2 (SPI module clock cycles)

The previous value of the CSHOLD bit in the SPI transmit data register (SPIDAT1) must be cleared to 0 for the C2T delay to be enabled.

**Note**—If the SPIDAT1.CSHOLD bit is set within the control field, the current hold time and the following setup time will not be applied in between transaction.

#### 2.5.1.2 Chip Select Hold Time

The master can be configured to provide a (slow) slave device a certain chip select hold time after the last edge on SPICLK. This delay is controlled by the T2CDELAY bit in the SPI delay register (SPIDELAY) and can be configured between for 1 and 256 SPI module clock cycles. The T2CDELAY is applicable only in 4-pin with chip select mode. The T2CDELAY begins after the data shifting period ends. The T2C delay period is specified by:

Maximum duration of T2CDELAY period = SPIDELAY.T2CDELAY + 1 (SPI module clock cycle)

If the PHASE bit in the SPI data format register n (SPIFMTn) is 0, then the T2CDELAY period lasts for an additional ½ SPICLK time over that specified by the above equation.

The current value of the CSHOLD bit in the SPI transmit data register (SPIDAT1) must be cleared to 0 for T2C delay to be enabled.

**Note**—If the SPIDAT1.CSHOLD bit is set within the control field, the current hold time and the following setup time will not be applied in between transaction.

### 2.5.1.3 Automatic Delay Between Transfers

The SPI master can automatically insert a delay of between 2 and 65 SPI module clock cycles between transmissions. This delay is controlled by the WDELAY field in the SPI data format register n (SPIFMTn) and is enabled by setting the WDEL bit in the SPI transmit data register (SPIDAT1) to 1. The WDELAY period begins when the T2EDELAY period terminates (if T2E delay period is enabled) or when the T2CDELAY period terminates (if T2E delay period was disabled and T2C delay period was enabled) or when the master deasserts $\overline{\text{SPISCS[n]}}$ (if T2E and T2C delay periods are disabled). If a transfer is initiated by writing a 32-bit value to SPIDAT1, then the new values of SPIDAT1.WDEL and SPIFMTn.WDELAY are used; otherwise, the old values of SPIDAT1.WDEL and SPIFMTn.WDELAY are used. The WDELAY delay period is specified by:

Maximum duration of WDELAY period = SPIFMTn.WDELAY + 2 (SPI module clock cycles)

### 2.5.1.4 Chip Select Hold Option

There are slave devices available that require the chip select signal to be held continuously active during several consecutive data word transfers. Other slave devices require the chip select signal to be deactivated between consecutive data word transfers. The SPI can support both types of slave devices. The CSHOLD bit in the SPI transmit data register (SPIDAT1) selects between the two options.

If the chip select hold option is enabled, the chip select will not toggle between two consecutive accesses. Therefore, the SPIDELAY.T2CDELAY of the first transfer and the SPIDELAY.C2TDELAY of the second transfer will not be applied. However, the wait delay could still be applied between the two transactions, if the WDEL bit in SPIDAT1 is set to 1.

When the CSHOLD bit is 0, during the data transmission, the value of the chip select number field (CSNR[n:0]) in the SPIDAT1 register is put on the chip select $\overline{\text{SPISCS[n]}}$ to $\overline{\text{SPISCS[0]}}$ pins. When the transmission finishes, the chip select default pattern (CSDEF[n:0]) is put on the $\overline{\text{SPISCS[n]}}$ to $\overline{\text{SPISCS[0]}}$ pins.

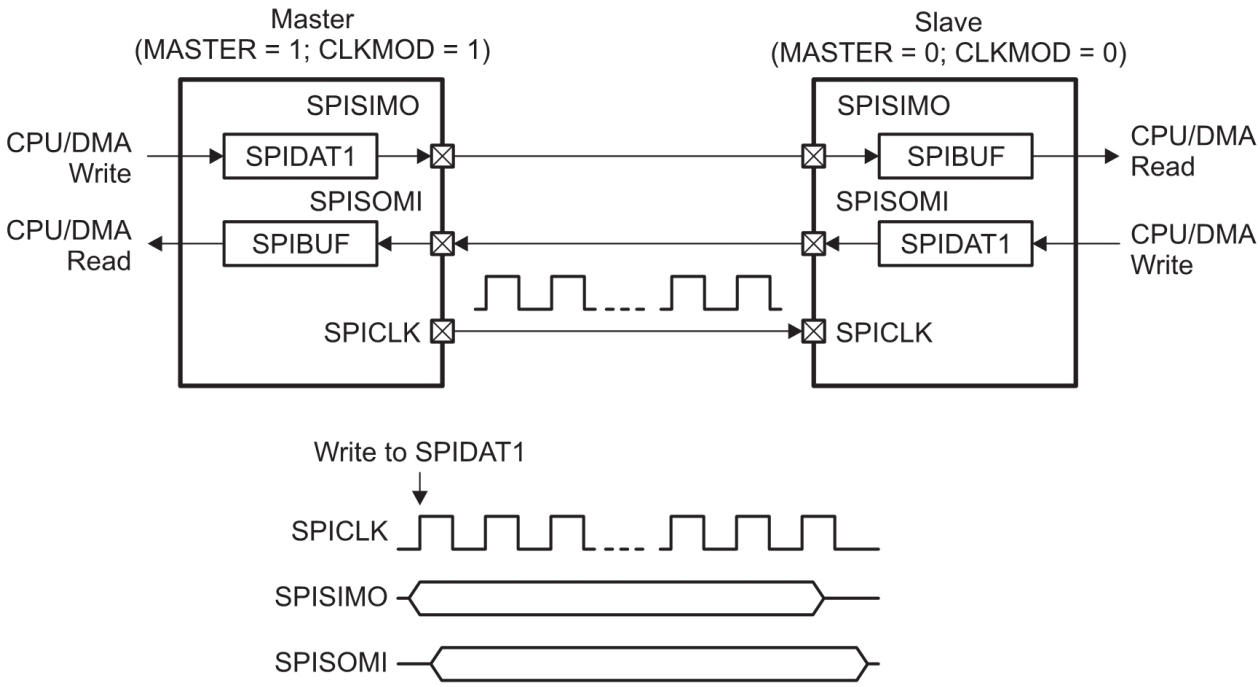
The current and previous values of the CSHOLD bit are retained. Although the current value of the CSHOLD bit is initialized to 0 when the RESET bit in the SPI global control register 0 (SPIGCR0) is cleared to 0, the previous value of the CSHOLD bit is not initialized. The previous value of the CSHOLD bit must be explicitly initialized by writing twice to the CSHOLD bit.

## 2.6 SPI Operation: 3-Pin Mode

**Note**—If only unidirectional communication is required, the SPICLK pin and the two data pins (SPISOMI and SPISIMO) must all be configured as functional pins. A 2-pin unidirectional mode is not supported.

The SPI 3-pin mode uses only the clock (SPICLK) and data (SPISOMI and SPISIMO) pins for bidirectional communication between master and slave devices. Figure 2-1 shows the basic 3-pin SPI option.

**Figure 2-1 SPI 3-Pin Option**




To select the 3-pin SPI option, the SPICLK, SPISOMI, and SPISIMO pins should be configured as functional pins by configuring the SPI pin control register 0 (SPIPC0).

The SPI operates in master mode only. The CLKMOD and MASTER bits in the SPI global control register 1 (SPIGCR1) must be programmed to 1 to configure the SPI for master mode. The SPI bus master is the device that drives the SPICLK signal and initiates SPI bus transfers. In SPI master mode, the SPISOMI pin output buffer is in a high-impedance state and the SPICLK and the SPISIMO pin output buffer is enabled.

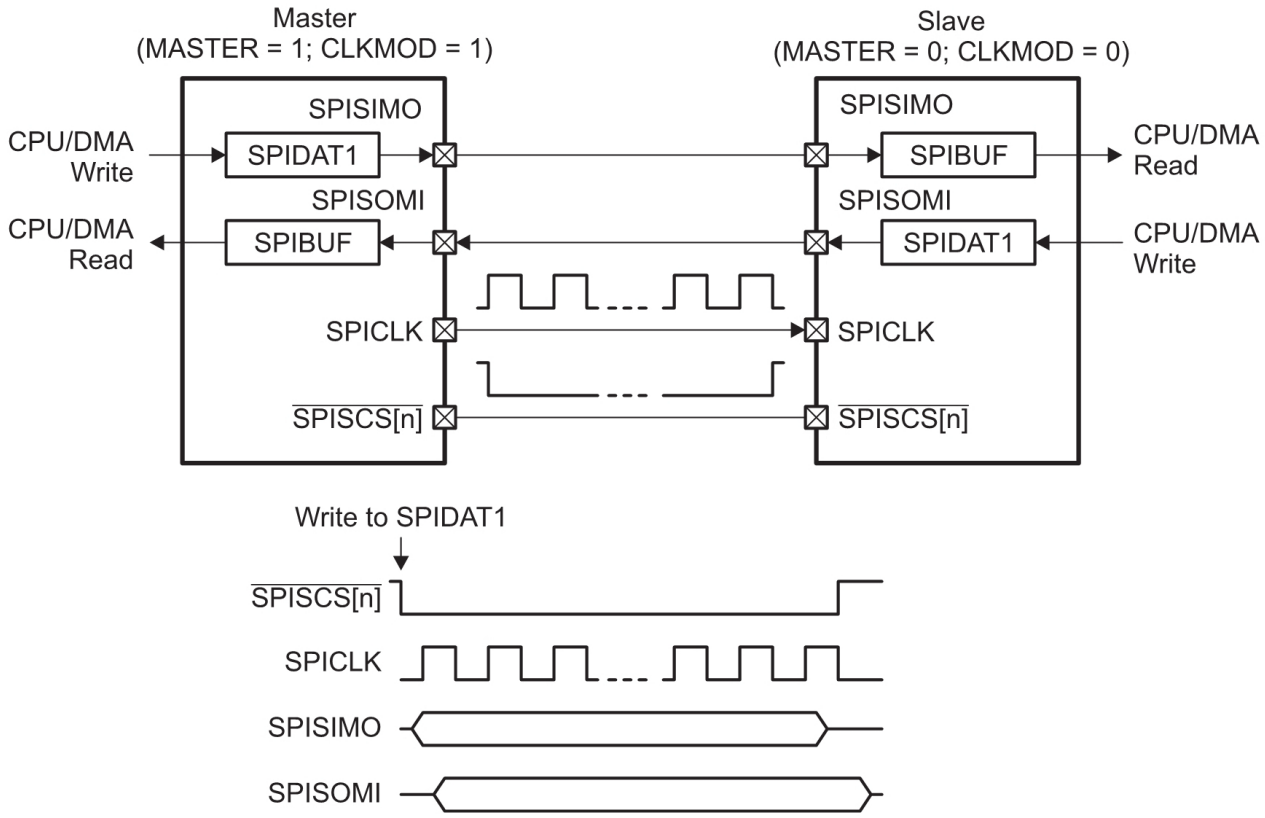
In master mode with the 3-pin option, the DSP writes transmit data to the SPI transmit data registers (SPIDAT0[15:0] or SPIDAT1[15:0]). This initiates a transfer. A series of clocks pulses will be driven out on the SPICLK pin to complete the transfer. Each clock pulse on the SPICLK pin causes the simultaneous transfer (in both directions) of one bit by both the master and slave SPI devices. CPU writes to the configuration bits in SPIDAT1 (not writing to SPIDAT1[15:0]) do not result in a new transfer. When the selected number of bits has been transmitted, the received data is transferred to the SPI receive buffer register (SPIBUF) for the CPU to read. Data is stored right-justified in SPIBUF.

**Note**—Either SPIDAT0 or SPIDAT1 can be used on both master and slaves sides.

## 2.7 SPI Operation: 4-Pin with Chip Select Mode

The 4-pin with chip select option is a superset of the 3-pin option and uses the chip select (SPISCS[n]) pin in addition to the clock (SPICLK) and data (SPISOMI and SPISIMO) pins.

**Figure 2-2 SPI 4-Pin Option with SPISCS[n]**




**Note—**Either SPIDAT0 or SPIDAT1 can be used on both master and slaves sides.

To select the 4-pin with chip select option, the SPICLK, SPISOMI, SPISIMO, and SPISCS[n] pins should be configured as functional pins by configuring the SPI pin control register 0 (SPIPC0).

In SPI master mode, the SPISOMI pin output buffer is in a high-impedance state and the SPICLK, SPISIMO, and SPISCS[n] pin output buffer is enabled.

In master mode, the SPISCS[n] pin functions as an output, and toggles when a specific slave device is selected. However, this is most useful on devices that support multiple SPISCS[n] pins. The SPI supports only a single SPISCS[n] and so the usefulness of this pin in master mode is limited. In practice, general-purpose I/O pins are needed to support multiple slave device chip selects.

However, one reason to use the SPISCS[n] pin as a functional pin for the SPI master is to take advantage of the timing parameters that can be set using the SPI delay register (SPIDELAY). The SPIDELAY allows delays to be added automatically so that the slave timing requirements between clock and chip select may be more easily met. Another reason would be to make use of the error detection built into the SPI.

## 2.8 Data Formats

The SPI provides the capability to configure four independent data formats. These formats are configured by programming the corresponding SPI data format registers (SPIFMTn). In each data format, the following characteristics of the SPI operation are selected:

- Character length from 2 to 16 bits: The character length is configured by the SPIFMTn.CHARLEN field.
- Shift direction (MSB first or LSB first): The shift out direction is configured by the SPIFMTn.SHIFTDIR bit.
- Clock polarity: The clock polarity is configured by the SPIFMTn.POLARITY bit.
- Clock phase: The clock phase is configured by the SPIFMTn.PHASE bit.

The data format is chosen on each transaction. Transmit data is written to the SPI transmit data register 1 (SPIDAT1) and in the same write the data word format select (DFSEL) bit in SPIDAT1 indicates which data format is to be used for the next transaction. Alternatively, the data format can be configured once and applies to all transactions that follow until the data format is changed.

### 2.8.1 Character Length

The character length is configured by the SPIFMTn.CHARLEN bit. Legal values are 2 bits (2h) to 16 bits (10h). The character length is independently configured for each of the four data formats; and it must be programmed in the master mode.

Transmit data is written to SPIDAT1. The transmit data must be written right-justified irrespective of the character length. The SPI automatically sends out the data correctly based on the chosen data format.

Figure 2-3 shows how a 12-bit word (EC9h) must be written to the transmit buffer in order to be transmitted correctly.

**Figure 2-3 Format for Transmitting 12-Bit Word**

| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| x | x | x | x | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

The data received in SPIBUF is right-justified irrespective of the character length and is padded with 0s when character length is less than 16.

Figure 2-4 shows how a 10-bit word (3A2h) is stored in the buffer once it is received.

**Figure 2-4 Format for 10-Bit Received Word**

| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

### 2.8.2 Shift Direction

The shift out direction is configured as most-significant bit (MSB) first or least significant bit (LSB) first. The SPI supports automatic right-alignment of receive data independent of shift direction and data word length. Transmit data has to be written right-aligned into the SPI and the internal shift register will sort out according to selected shift direction and data word length for correct transfer.

The shift out direction is selected by the SPIFMTn.SHIFTDIR bit. The shift out direction is independently configured for each of the four data formats.

- When SPIFMTn.SHIFTDIR is 0, the transmit data is shifted out MSB first.
- When SPIFMTn.SHIFTDIR is 1, the transmit data is shifted out LSB first.

### 2.8.3 Clock Phase and Polarity

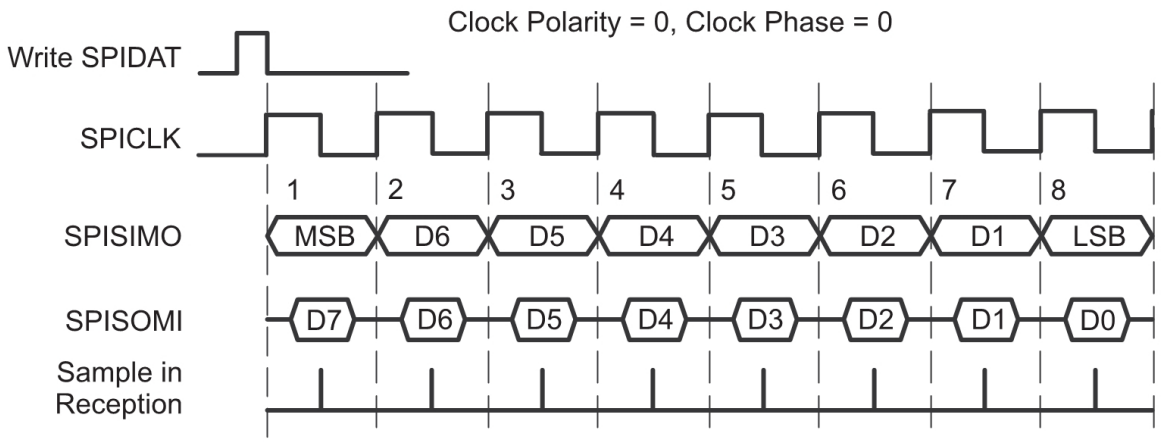
The SPI provides the flexibility to program four different clock mode combinations that SPICLK may operate, enabling a choice of the clock phase (delay or no delay) and the clock polarity (rising edge or falling edge). When operating with PHASE active, the SPI makes the first bit of data available after SPIDAT1 is written and before the first edge of SPICLK. The data input and output edges depend on the values of both the POLARITY and PHASE bits as shown in Table 2-5.

**Table 2-5 Clocking Modes**

| POLARITY | PHASE | Action |
|---|---|---|
| 0 | 0 | Data is output on the rising edge of SPICLK. Input data is latched on the falling edge. |
| 0 | 1 | Data is output one half-cycle before the first rising edge of SPICLK and on subsequent falling edges. Input data is latched on the rising edge of SPICLK. |
| 1 | 0 | Data is output on the falling edge of SPICLK. Input data is latched on the rising edge. |
| 1 | 1 | Data is output one half-cycle before the first falling edge of SPICLK and on subsequent rising edges. Input data is latched on the falling edge of SPICLK. |

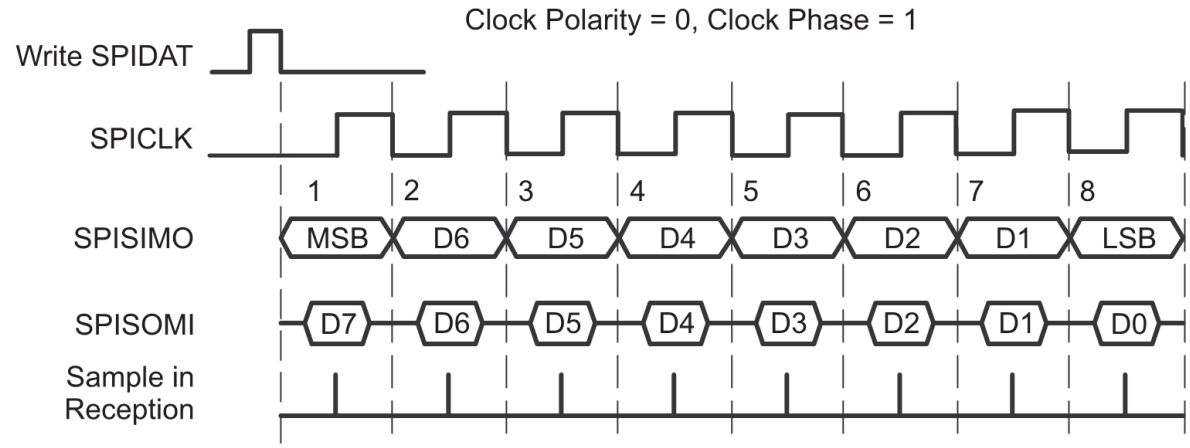
Figure 2-5 to Figure 2-8 show the four possible signals of SPICLK corresponding to each mode. Having four signal options allows the SPI to interface with different types of serial devices. Also shown are the SPICLK control bit polarity and phase values corresponding to each signal.

**Figure 2-5 Clock Mode with POLARITY = 0 and PHASE = 0 (A)**




(A) Clock phase = 0 (SPICLK without delay)
- » Data is output on the rising edge of SPICLK.
- » Input data is latched on the falling edge of SPICLK.
- » A write to the SPIDAT register starts SPICLK.

**Figure 2-6 Clock Mode with POLARITY = 0 and PHASE = 1 (A)**

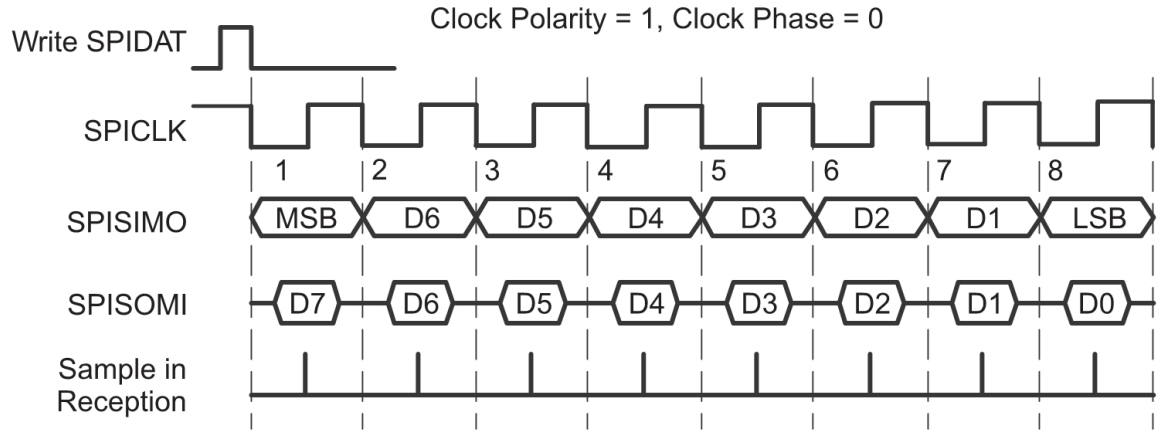



(A) Clock phase = 1 (SPICLK with delay)

» Data is output one-half cycle before the first rising of SPICLK and on subsequent falling edges of SPICLK.

» Input data is latched on the rising edge of SPICLK.

**Figure 2-7 Clock Mode with POLARITY = 1 and PHASE = 0 (A)**



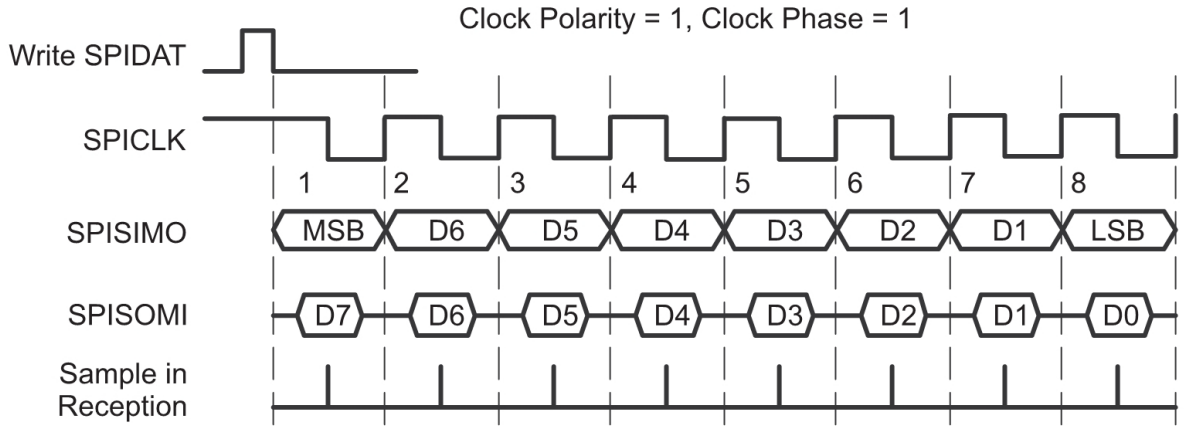

(A) Clock phase = 0 (SPICLK without delay)

» Data is output on the falling edge of SPICLK.

» Input data is latched on the rising edge of SPICLK.

» A write to the SPIDAT register starts SPICLK.

**Figure 2-8 Clock Mode with POLARITY = 1 and PHASE = 1 (A)**



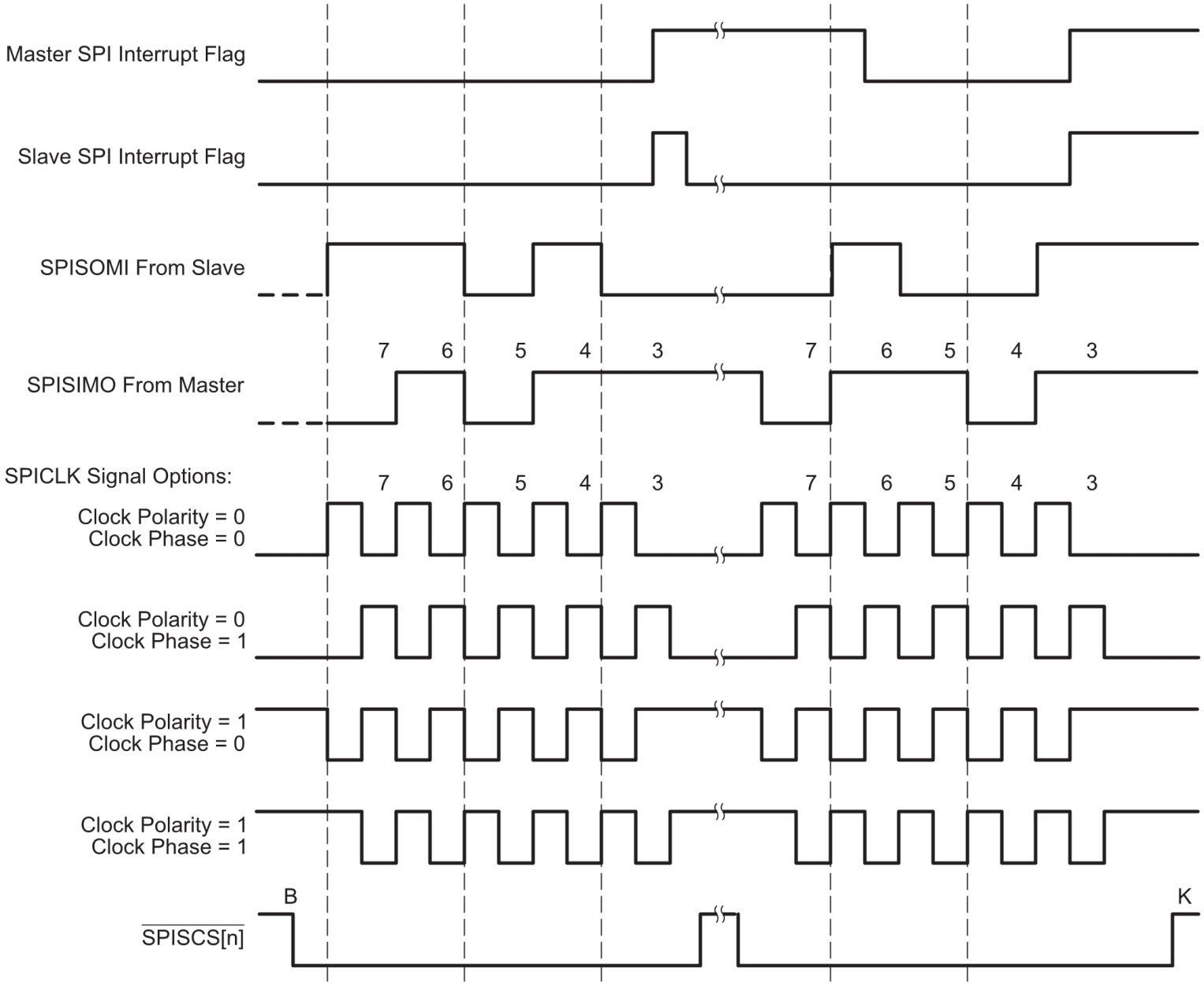

(A) Clock phase = 1 (SPICLK with delay)

» Data is output one-half cycle before the first falling edge of SPICLK and on the subsequent rising edges of SPICLK.

» Input data is latched on the falling edge of SPICLK.

## 2.8.4 SPI Data Transfer Example

**Figure 2-9 SPI Data Transfer, Five Bits per Character (4-Pin with Chip Select Option)**



## 2.9 Interrupt Support

The SPI module outputs two interrupts that are routed to the CPU. The SPI interrupt system is controlled by three registers:

- The SPI interrupt level register (SPILVL) controls which events (INT0 or INT1) are assigned to each interrupt.
- The SPI interrupt register (SPIINT0) contains bits to selectively enable/disable each interrupt event.
- The SPI flag register (SPIFLG) contains flags indicating when each of the interrupt conditions have occurred.

Multiple interrupt sources can be assigned to the same CPU interrupt. To identify the interrupt source in the SPI peripheral, the CPU reads the SPI flag status register (SPIFLG) or the INTVECTn code in the SPI interrupt vector register n (INTVECn).

Check the device-specific data manual for details on the exact CPU interrupt numbers assigned to the SPI interrupts.

## 2.10 DMA Events Support

If handling the SPI message traffic on a character-by-character basis requires too much CPU overhead, then the CPU can configure the system DMA to handle the SPI data transfer.

The SPI module has two DMA synchronization event outputs for receive (REVT) and transmit (XEVT), allowing DMA transfers to be triggered by SPI read receive and write transmit events. The SPI module enables DMA requests by enabling the DMA request enable (DMAREQEN) bit in the SPI interrupt register (SPIINT0).

When a character is to be transmitted the SPI module signals the DMA via the XEVT signal. The DMA controller then transfers the data from the source buffer into the SPI transmit data register (SPIDAT1). When a character is received, the SPI module signals the DMA via the REVT signal. The DMA controller then reads the data from the SPI receive buffer register (SPIBUF) and transfers it to a destination buffer for ready access.

In most cases, if the DMA is being used to service received data from the SPI, the receive interrupt enable (RXINTEN) bit in SPIINT0 should be cleared to 0. This prevents the CPU from responding to the received data in addition to the DMA. For specific SPI synchronization event number assignments and detailed DMA features, see your device-specific data manual.

## 2.11 Robustness Features

The SPI module includes many features to make the SPI communication link robust. A internal loopback test mode can be used to facilitate a power on self test routine. Additionally, the SPI master continually monitors the bus for faults on its data line. The following sections describe these robustness features in more detail.

### 2.11.1 SPI Internal Loopback Test Mode (Master Only)

**CAUTION—**The internal loop-back self-test mode should not be entered during a normal data transaction or unpredictable operation may occur.

To select the loopback mode, the SPICLK, SPISOMI, SPISIMO pins should be configured as functional pins by configuring the SPI pin control register 0 (SPIPC0) and by setting the LOOPBACK bit in the SPI global control register 1 (SPIGCR1). The internal loop-back self-test mode can be used to test the SPI transmit path and receive path including the transmit and receive buffers. In this mode, the transmit signal is internally fed back to the receiver and the SPISIMO, SPISOMI, and SPICLK pins are in a high-impedance state. This mode allows the CPU to write into the transmit buffer, and check that the receive buffer contains the correct transmit data. If an error occurs the corresponding error is set within the status field.

### 2.11.2 SPI Transmission Continuous Self-Test

During a data transfer, the SPI inputs the value from its data output pin on the appropriate SPICLK edge. This value is compared against the expected value and any difference indicates a fault on the SPI bus. If a fault is detected, then the BITERR bit in the SPI receive buffer register (SPIBUF) and the BITERRFLG bit in the SPI flag register (SPIFLG) are set and an error interrupt is generated if enabled. The SPI continuous self-test mode is not available in SPI loopback mode.

## 2.12 Reset Considerations

This section describes the software and hardware reset considerations.

### 2.12.1 Software Reset Considerations

The SPI module contains a software reset (RESET) bit in the SPI global control register 0 (SPIGCR0) that is used to reset the SPI module. As a result of a reset, the SPI module register values go to their reset state. The RESET bit must be set before any operation on the SPI is done.

### 2.12.2 Hardware Reset Considerations

In the event of a hardware reset, the SPI module register values go to their reset state and the application software needs to reprogram the registers to the desired values.

## 2.13 Power Management

The SPI module can be put in either local or global low-power mode. Global low-power mode is asserted by the system and is not controlled by the SPI. During global low-power mode, all clocks to the SPI are turned off so the module is completely inactive.

The SPI local low-power mode is asserted by setting the POWERDOWN bit in the SPI global control register 1 (SPIGCR1). Setting this bit stops the clocks to the SPI internal logic and the SPI registers. Setting the POWERDOWN bit causes the SPI to enter local low-power mode and clearing the POWERDOWN bit causes SPI to exit from local low-power mode. All the registers are accessible during local power-down mode as any register access enables the clock to SPI for that particular access alone.

Because entering a low-power mode has the effect of suspending all state machine activities, care must be taken when entering such modes to ensure that a valid state is entered when low-power mode is active. As a result, application software must ensure that a low-power mode is not entered during a transmission or reception of data.

## 2.14 Emulation Considerations

**CAUTION—**Viewing or otherwise reading the following SPI registers: SPIBUF, SPIFLG, INTVEC0, and INTVEC1 through the JTAG debugger will cause their contents to change, possibly invalidating the results of the debug session. Be sure to set up the debugger to avoid reading these registers.

The SPI module does not support soft or hard stop during emulation breakpoints. The SPI module will continue to run if an emulation breakpoint is encountered.

In addition, any status registers that are cleared after reading will be affected if viewed in a memory or watch window of the debugger because the emulator will read these registers to update the value displayed in the window.

## 2.15 Initialization

Perform the following procedure for initializing the SPI:

1. Reset the SPI by clearing the RESET bit in the SPI global control register 0 (SPIGCR0) to 0.
2. Take the SPI out of reset by setting SPIGCR0.RESET to 1.
3. Configure the SPI for master mode by configuring the CLKMOD and MASTER bits in the SPI global control register 1 (SPIGCR1).
4. Configure the SPI for 3-pin or 4-pin with chip select mode by configuring the SPI pin control register 0 (SPIPC0).
5. Choose the SPI data format register n (SPIFMTn) to be used by configuring the DFSEL bit in the SPI transmit data register (SPIDAT1).
6. Configure the SPI data rate, character length, shift direction, phase, polarity and other format options using SPIFMTn selected in step 5.
7. In master mode, configure the master delay options using the SPI delay register (SPIDELAY).
8. Select the error interrupt notifications by configuring the SPI interrupt register (SPIINT0) and the SPI interrupt level register (SPILVL).
9. Enable the SPI communication by setting the SPIGCR1.ENABLE to 1.
10. Setup and enable the DMA for SPI data handling and then enable the DMA servicing for the SPI data requests by setting the SPIINT0.DMAREQEN to 1.
11. Handle SPI data transfer requests using DMA and service any SPI error conditions using the interrupt service routine.

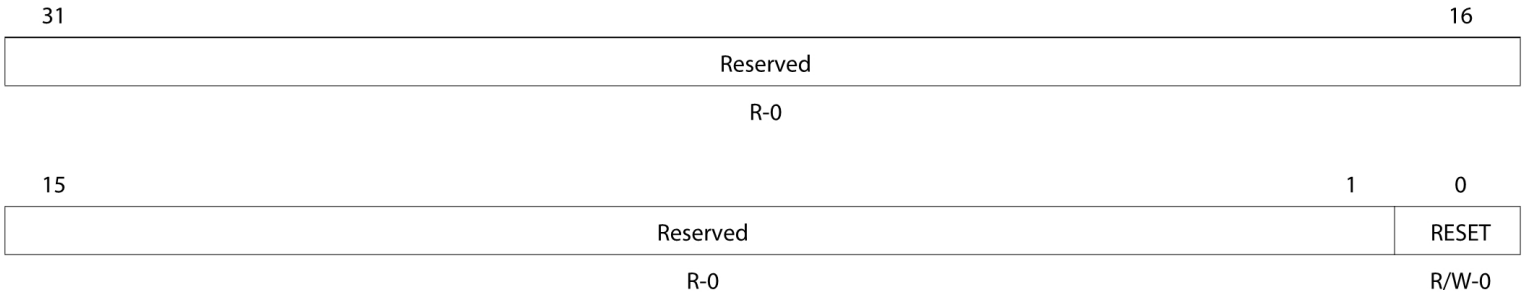
**Table 3-1 SPI Registers**

| Offset Address [1] | Acronym | Register Description | Section |
|---|---|---|---|
| 0h | SPIGCR0 | SPI Global Control Register 0 | Section 3.1 |
| 4h | SPIGCR1 | SPI Global Control Register 1 | Section 3.2 |
| 8h | SPIINT0 | SPI Interrupt Register | Section 3.3 |
| Ch | SPILVL | SPI Interrupt Level Register | Section 3.4 |
| 10h | SPIFLG | SPI Flag Register | Section 3.5 |
| 14h | SPIPC0 | SPI Pin Control Register 0 (Function) | Section 3.6 |
| 38h | SPIDAT0 | SPI Data Transmit Register 0 | Section 3.7 |
| 3Ch | SPIDAT1 | SPI Data Transmit Register 1 (Data Transmit and Format Select) | Section 3.8 |
| 40h | SPIBUF | SPI Receive Buffer Register | Section 3.9 |
| 44h | SPIEMU | SPI Receive Emulation Register | Section 3.10 |
| 48h | SPIDELAY | SPI Delay Register | Section 3.11 |
| 4Ch | SPIDEF | SPI Default Chip Select Register | Section 3.12 |
| 50h | SPIFMT0 | SPI Data Format Register 0 | Section 3.13 |
| 54h | SPIFMT1 | SPI Data Format Register 1 | Section 3.13 |
| 58h | SPIFMT2 | SPI Data Format Register 2 | Section 3.13 |
| 5Ch | SPIFMT3 | SPI Data Format Register 3 | Section 3.13 |
| 60h | INTVEC0 | SPI Interrupt Vector Register 0 | Section 3.14 |
| 64h | INTVEC1 | SPI Interrupt Vector Register 1 | Section 3.15 |

1. The actual addresses of these registers are device specific. See your device-specific data manual to verify the SPI register addresses.

## 3.1 SPI Global Control Register 0 (SPIGCR0)

**Figure 3-1 SPI Global Control Register 0 (SPIGCR0)**




Legend: R = Read only; R/W = Read/Write; -*n* = value after reset

**Table 3-2 SPI Global Control Register 0 (SPIGCR0) Field Descriptions**

| Bit | Field | Description |
|---|---|---|
| 31-1 | Reserved | Reads return 0 and writes have no effect. |
| 0 | RESET | Reset bit for the module. This bit needs to be set to 1 before any operation on SPI can be done.<br>0 = SPI is in reset state.<br>1 = SPI is out of reset state. |

## 3.2 SPI Global Control Register 1 (SPIGCR1)

**Figure 3-2 SPI Global Control Register 1 (SPIGCR1)**

| 31 | 25 | 24 | 23 | 17 | 16 |
|---|---|---|---|---|---|
| Reserved | | ENABLE | Reserved | | LOOPBACK |
| R-0 | | R/W-0 | R-0 | | R/W-0 |

| 15 | 9 | 8 | 7 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| Reserved | | POWERDOWN | Reserved | | CLKMOD | MASTER |
| R-0 | | R/W-0 | R-0 | | R/W-0 | R/W-0 |

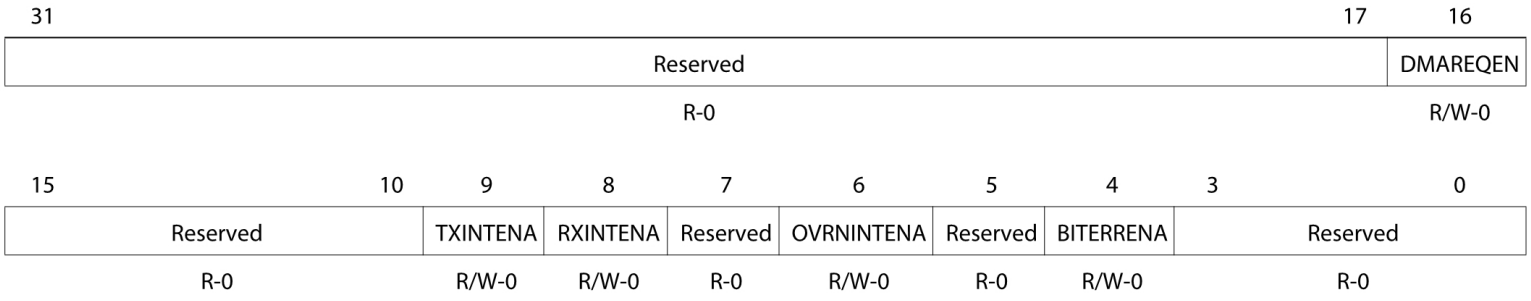
Legend: R = Read only; R/W = Read/Write; -*n* = value after reset

**Table 3-3 SPI Global Control Register 1 (SPIGCR1) Field Descriptions**

| Bit | Field | Description |
|---|---|---|
| 31-25 | Reserved | Reads return 0 and writes have no effect. |
| 24 | ENABLE | SPI enable. This bit enables the SPI transfers. The other SPI configuration registers except SPIINT0.DMAREQEN should be configured before writing a 1 to this bit. This will prevent the SPI from responding to bus operations erroneously while it is in the process of being configured. The SPIINT0.DMAREQEN should be enabled after setting ENABLE. If SPIINT0.DMAREQEN is enabled before setting ENABLE then the first DMA request that occurs before the SPI is ready for data transfer may get dropped.<br>When ENABLE bit is cleared to 0, the following SPI registers get forced to their default states (to 0s except for RXEMPTY bit in SPIBUF):<br>• Both TX and RX shift registers<br>• The TXDATA fields of SPIDAT0 and SPIDAT1 registers<br>• All the fields of the SPIFLG register<br>• Contents of SPIBUF and the internal RXBUF registers<br>0 = SPI is not activated for transfers.<br>1 = Activates SPI. |
| 23-17 | Reserved | Reads return 0 and writes have no effect. |
| 16 | LOOPBACK | Internal loop-back test mode. The internal self-test option can be enabled by setting this bit. If the SPISIMO and SPISOMI pins are configured with SPI functionality, then the SPISIMO pin is internally connected to the SPISOMI pin. The transmit data is looped back as receive data and is stored in the receive field of the concerned buffer.<br>Externally, during loop-back operation, the SPICLK pin outputs an inactive value, SPISIMO and SPISOMI pins remain in high-impedance state. The SPI has to be initialized in master mode before the loop-back can be selected. If the SPI is initialized in slave mode or a data transfer is ongoing, errors may result.<br>0 = Internal loop-back test mode disabled.<br>1 = Internal loop-back test mode enabled. |
| 15-9 | Reserved | Reads return 0 and writes have no effect. |
| 8 | POWERDOWN | When active, the SPI state machine enters a power-down state.<br>0 = The SPI is in active mode.<br>1 = The SPI is in power-down mode. |
| 7-2 | Reserved | Reads return 0 and writes have no effect. |
| 1-0 | CLKMOD, MASTER | These two bits (CLKMOD, MASTER) determine whether the SPI operates in master or slave mode.<br>0-2h = Reserved<br>3h = MASTER MODE. SPICLK is an output and the SPI initiates transfers. Data is transmitted on the SPISIMO pin and received on the SPISOMI pin. The $\overline{\text{SPISCS[n]}}$ pin is an output pin if configured as SPI slave chip select.<br>**Note:** Both CLKMOD and MASTER bits are retained for compatibility purposes even though only one value (3h) is valid for master operation of SPI |

## 3.3 SPI Interrupt Register (SPIINT0)

**Figure 3-3 SPI Interrupt Register (SPIINT0)**



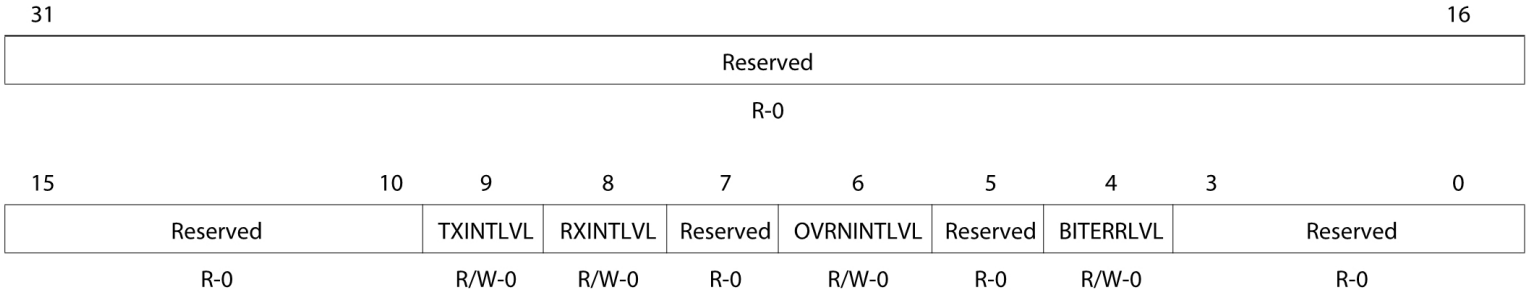

Legend: R = Read only; W = Write only; -*n* = value after reset

**Table 3-4 SPI Interrupt Register (SPIINT0) Field Descriptions**

| Bit | Field | Description |
|---|---|---|
| 31-17 | Reserved | Reads return 0 and writes have no effect. |
| 16 | DMAREQEN | DMA request enable. Enables the DMA request signal to be generated for both receive and transmit channels. Set DMAREQEN only after setting the SPIGCR1.ENABLE bit to 1.<br>0 = DMA is not used.<br>1 = DMA requests will be generated.<br>**Note:** A transmit DMA request will be generated each time a transmit data is copied to the shift register either from TXBUF or directly from SPIDAT0/SPIDAT1.<br>**Note:** A receive DMA request will be generated each time a received data is copied to SPIBUF register either from RXBUF or directly from the shift register. |
| 15-10 | Reserved | Reads return 0 and writes have no effect. |
| 9 | TXINTENA | An interrupt is to be generated every time data is written to the shift register, so that a new data can be written to TXBUF. Setting this bit will generate an interrupt if the SPIFLG.TXINTFLG bit is set to 1.<br>0 = No interrupt will be generated upon SPIFLG.TXINTFLG being set to 1.<br>1 = Interrupt will be generated upon SPIFLG.TXINTFLG being set to 1. |
| 8 | RXINTENA | Receive interrupt enable. An interrupt is to be generated when the SPIFLG.RXINTFLAG bit is set.<br>0 = Interrupt will not be generated.<br>1 = Interrupt will be generated. |
| 7 | Reserved | Reads return 0 and writes have no effect. |
| 6 | OVRNINTENA | Overrun interrupt enable. An interrupt is to be generated when the SPIFLG.OVRNINTFLG bit is set.<br>0 = Overrun interrupt will not be generated.<br>1 = Overrun interrupt will be generated. |
| 5 | Reserved | Reads return 0 and writes have no effect. |
| 4 | BITERRENA | Enables interrupt on bit error. An interrupt is to be generated when the SPIFLG.BITERRFLG is set.<br>0 = No interrupt asserted upon bit error.<br>1 = Enables an interrupt on a bit error. |
| 3-0 | Reserved | Reads return 0 and writes have no effect. |

## 3.4 SPI Interrupt Level Register (SPILVL)

**Figure 3-4 SPI Interrupt Level Register (SPILVL)**



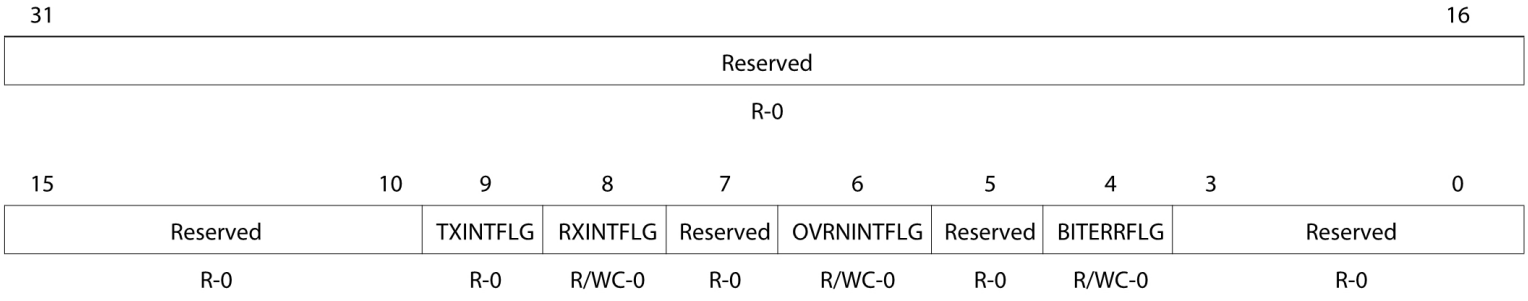

Legend: R = Read only; W = Write only; -*n* = value after reset

**Table 3-5 SPI Interrupt Level Register (SPILVL) Field Descriptions**

| Bit | Field | Description |
|---|---|---|
| 31-10 | Reserved | Reads return 0 and writes have no effect. |
| 9 | TXINTLVL | Transmit interrupt level.<br>0 = Transmit interrupt is mapped to interrupt line INT0.<br>1 = Transmit interrupt is mapped to interrupt line INT1. |
| 8 | RXINTLVL | Receive interrupt level.<br>0 = Receive interrupt is mapped to interrupt line INT0.<br>1 = Receive interrupt is mapped to interrupt line INT1. |
| 7 | Reserved | Reads return 0 and writes have no effect. |
| 6 | OVRNINTLVL | Receive overrun interrupt level.<br>0 = Receive overrun interrupt is mapped to interrupt line INT0.<br>1 = Receive overrun interrupt is mapped to interrupt line INT1. |
| 5 | Reserved | Reads return 0 and writes have no effect. |
| 4 | BITERRLVL | Bit error interrupt level.<br>0 = Bit error interrupt is mapped to interrupt line INT0.<br>1 = Bit error interrupt is mapped to interrupt line INT1. |
| 3-0 | Reserved | Reads return 0 and writes have no effect. |

## 3.5 SPI Flag Register (SPIFLG)

**Figure 3-5 SPI Flag Register (SPIFLG)**




Legend: R = Read only; W = Write only, WC = Write/Clear; -*n* = value after reset

**Table 3-6 SPI Flag Register (SPIFLG) Field Descriptions (Part 1 of 2)**

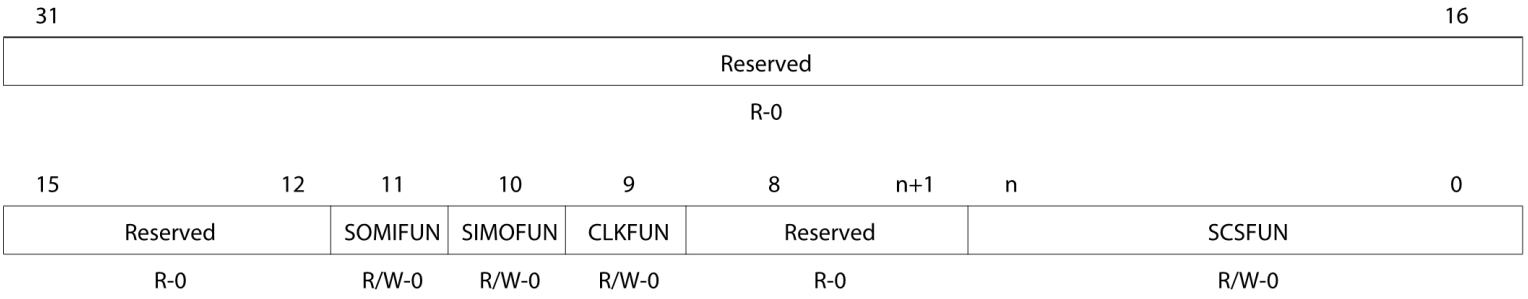
| Bit | Field | Description |
|---|---|---|
| 31-10 | Reserved | Reads return 0 and writes have no effect. |
| 9 | TXINTFLG | Transmitter empty interrupt flag. Serves as an interrupt flag indicating that the transmit buffer (TXBUF) is empty and a new data can be written to it. This flag is set when a data is copied to the shift register either directly or from the TXBUF register. This bit is cleared by one of following ways:<br>• Writing a new data to either SPIDAT0 or SPIDAT1<br>• Writing a 0 to SPIGCR1.ENABLE<br>0 = Transmit buffer is now full. No interrupt pending for transmitter empty.<br>1 = Transmit buffer is empty. An interrupt is pending to fill the transmitter. |
| 8 | RXINTFLG | Receiver full interrupt flag. This flag is set when a word is received and copied into the buffer register (SPIBUF). This bit is cleared under the following ways:<br>• Reading the SPIBUF register. During emulation mode, however, a read to the emulation register (SPIEMU) does not clear this flag bit.<br>• Reading INTVEC0 or INTVEC1 register when there is a receive buffer full interrupt<br>• Writing a 1 to this bit<br>• Writing a 0 to SPIGCR1.ENABLE<br>• System reset<br>0 = No new received data pending. Receive buffer is empty.<br>1 = A newly received data is ready to be read. Receive buffer is full.<br>**Note:** Clearing RXINTFLG bit by writing a 1 before reading the SPIBUF sets the RXEMPTY bit of the SPIBUF register too. This way, one can ignore a received data. However, if the internal RXBUF is already full, the data from RXBUF will be copied to SPIBUF and the RXEMPTY bit will be cleared again. The SPIBUF contents should be read first if this situation needs to be avoided. |
| 7 | Reserved | Reads return 0 and writes have no effect. |
| 6 | OVRNINTFLG | Receiver overrun flag. The bit is set when a receive operation completes before the previous character has been read from the receive buffer. The bit indicates that the last received character has been overwritten and therefore lost. This bit is cleared under the following conditions:<br>• Reading INTVEC0 or INTVEC1 register when there is a receive buffer overrun interrupt<br>• Writing a 1 to this bit<br>0 = Overrun condition did not occur.<br>1 = Overrun condition has occurred.<br>**Note:** Reading SPIBUF register does not clear the OVRNINTFLG bit. If an overrun interrupt is detected, then the SPIBUF may need to be read twice to get to the overrun buffer. This is due to the fact that the overrun will always occur to the internal RXBUF. Each read to the SPIBUF will result in RXBUF contents (if it is full) getting copied to SPIBUF.<br>**Note:** A special condition under which OVRNINTFLG flag gets set. If both SPIBUF and RXBUF are already full and while another buffer receive is underway, if any errors like TIMEOUT, BITERR and DLENERR occur, then OVRNINTFLG will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a normal receiver overrun. |
| 5 | Reserved | Reads return 0 and writes have no effect. |

**Table 3-6 SPI Flag Register (SPIFLG) Field Descriptions (Part 2 of 2)**

| Bit | Field | Description |
|---|---|---|
| 4 | BITERRFLG | This bit is set when a mismatch of internal transmit data and transmitted data is detected. The SPI samples the signal of the transmit pin (master: SPISIMO, slave: SPISOMI) at the receive point (half clock cycle after transmit point). If the sampled value differs from the transmitted value a bit error is detected and the flag is set. A possible reason for a bit error can be a too high bit rate/capacitive load or another master/slave trying to transmit at the same time. This flag can be cleared by one of the following ways:<br>• Write a 1 to this bit.<br>• Set SPIGCR1.ENABLE bit to 0.<br>0 = No bit error occurred.<br>1 = A bit error occurred. |
| 3-0 | Reserved | Reads return 0 and writes have no effect. |

## 3.6 SPI Pin Control Register 0 (SPIPC0)

**Figure 3-6 SPI Pin Control Register 0 (SPIPC0)**



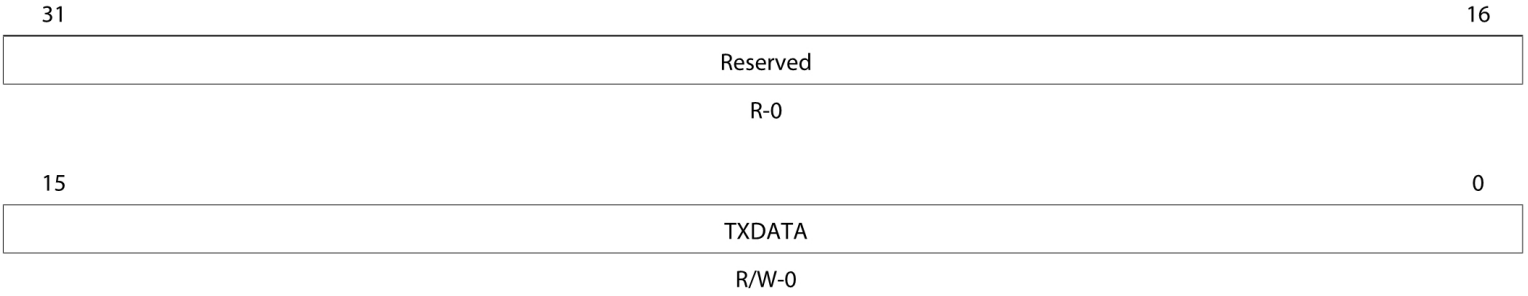

Legend: R = Read only; W = Write only; -*n* = value after reset

**Table 3-7 SPI Pin Control Register 0 (SPIPC0) Field Descriptions**

| Bit | Field | Description |
|---|---|---|
| 31-12 | Reserved | Reads return 0 and writes have no effect. |
| 11 | SOMIFUN | Slave out, master in pin function. This bit determines whether the SPISOMI pin is Reserved or a SPI functional pin.<br>0 = SPISOMI pin is reserved.<br>1 = SPISOMI pin is a SPI functional pin. |
| 10 | SIMOFUN | Slave in, master out pin function. This bit determines whether the SPISIMO pin is Reserved or a SPI functional pin.<br>0 = SPISIMO pin is reserved.<br>1 = SPISIMO pin is a SPI functional pin. |
| 9 | CLKFUN | SPI clock pin function. This bit determines whether the SPICLK pin is Reserved or a SPI functional pin.<br>0 = SPICLK pin is reserved.<br>1 = SPICLK pin is a SPI functional pin. |
| 8-(n+1) | Reserved | Reads return 0 and writes have no effect. |
| n-0 | SCSFUN | SPI chip select pin function. The bit position *i* in this field determines whether the $\overline{\text{SPISCS[i]}}$ pin is Reserved or a SPI functional pin. Please see the device-specific data manual for supported number of the chip select pins.<br>Bit position *i* in this field:<br>0 = The corresponding $\overline{\text{SPISCS[i]}}$ pin is Reserved<br>1 = The corresponding $\overline{\text{SPISCS[i]}}$ pin is a SPI functional pin. |

## 3.7 SPI Transmit Data Register 0 (SPIDAT0)

**Figure 3-7 SPI Data Register 0 (SPIDAT0)**



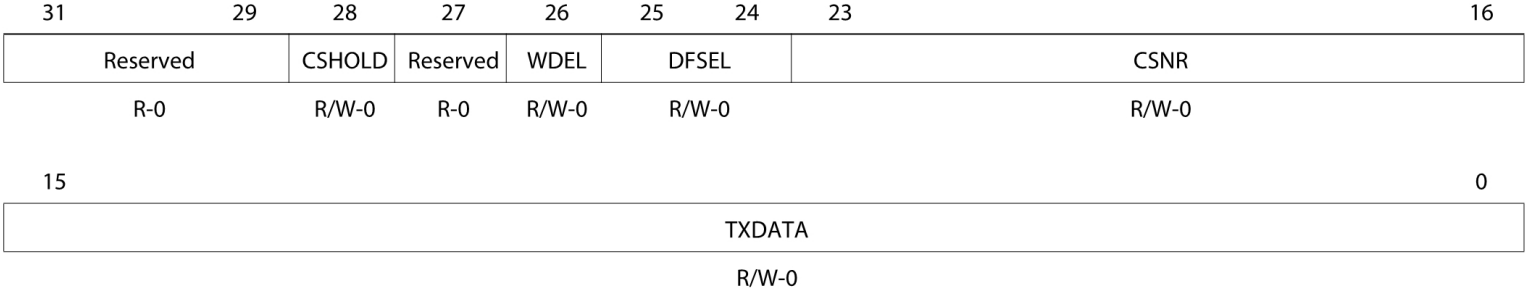

Legend: R = Read only; W = Write only; -*n* = value after reset

**Table 3-8 SPI Data Register 0 (SPIDAT0) Field Descriptions**

| Bit | Field | Description |
|---|---|---|
| 31-16 | Reserved | Reads return 0 and writes have no effect. |
| 15-0 | TXDATA | SPI transmit data (value = 0-FFFFh). When written, these bits will be copied to the shift register if it is empty. If the shift register is not empty, the TXBUF will hold the written values. SPIGCR1.ENABLE must be set to 1 before this register can be written to. Writing a 0 to the SPIGCR1.ENABLE forces the TXDATA[15:0] field to 0.<br>**Note:** Irrespective of the character length, the transmit data should be right-justified before writing to SPIDAT0 register.<br>**Note:** The default data format control register for SPIDAT0 is SPIFMT0. However, it is possible to reprogram the DFSEL field of SPIDAT1 before using SPIDAT0, to select a different SPIFMTn register. |

## 3.8 SPI Transmit Data Register 1 (SPIDAT1)

**Figure 3-8 SPI Data Register 1 (SPIDAT1)**



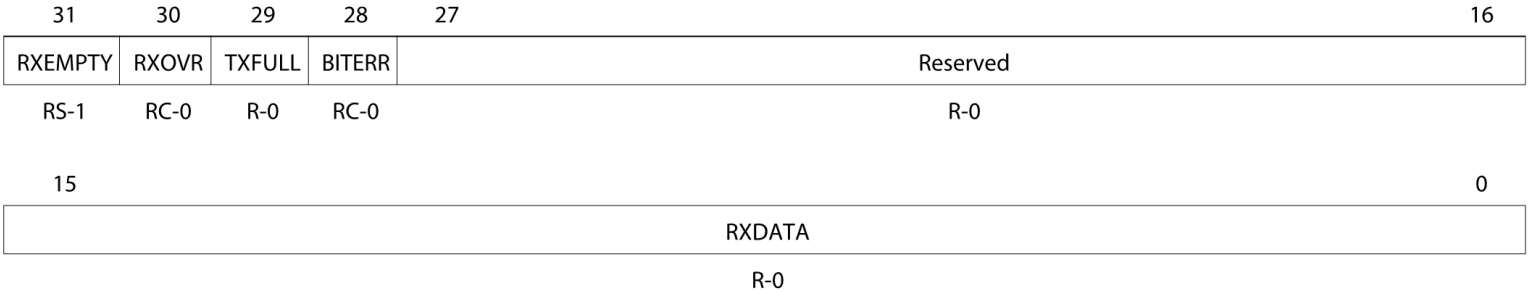

Legend: R = Read only; W = Write only; -*n* = value after reset

**Table 3-9 SPI Data Register 1 (SPIDAT1) Field Descriptions**

| Bit | Field | Description |
|---|---|---|
| 31-29 | Reserved | Reads return 0 and writes have no effect. |
| 28 | CSHOLD | Chip select hold mode. The CSHOLD bit is supported in master mode only. In slave mode, this bit is ignored. CSHOLD defines the behavior of the chip select lines at the end of a data transfer.<br>0 = The $\overline{\text{SPISCS[n]}}$ to $\overline{\text{SPISCS[0]}}$ pins are restored to CSDEF[n:0] bits in SPIDEF register at the end of a transfer after the T2CDELAY time has passed.<br>1 = The $\overline{\text{SPISCS[n]}}$ to $\overline{\text{SPISCS[0]}}$ pins are continued as CSNR[n:0] at the end of a transfer until a control field with new data and control information is loaded into SPIDAT1. The setup and hold delays in SPIDELAY register (C2TDELAY and C2TDELAY) are not applied between transactions which have CSHOLD=1.<br>**Note:** If a transfer with CSHOLD=0 is followed by a transfer with CSHOLD=1, then SPIDELAY.C2TDELAY will be applied at the beginning of the second transfer. To do multiple transactions to a slave without deactivating chip select between transfers, maintain CSHOLD of 1 and same CSNR information to keep the same slave selected in the transfers. |
| 27 | Reserved | Reads return 0 and writes have no effect. |
| 26 | WDEL | Enable the delay counter at the end of the current transaction. The WDEL bit is supported in master mode only. In slave mode, this bit is ignored.<br>0 = No delay will be inserted. However, $\overline{\text{SPISCS[n]}}$ pin will still be deactivated for at least 2 SPI module clock cycles if CSHOLD = 0.<br>1 = After a transaction, SPIFMTn.WDELAY of the selected data format will be loaded into the delay counter. No transaction will be performed until the SPIFMTn.WDELAY counter overflows. The $\overline{\text{SPISCS[n]}}$ pin will be deactivated for at least (WDELAY + 2) × SPI module clock period. |
| 25-24 | DFSEL | Data word format select:<br>0 = Data word format 0 is selected<br>1h = Data word format 1 is selected<br>2h = Data word format 2 is selected<br>3h = Data word format 3 is selected<br>**Note:** Preselecting a Format Register. Writing to just the control field (using byte writes) does not initiate any SPI transfer in master mode. This feature can be used to set up SPICLK phase or polarity before actually starting the transfer by just updating the DFSEL fields in the control field to select the required phase/polarity combination. |
| 23-16 | CSNR | Chip select number (value = 0-FFh). The CSNR field defines the chip select that will be activated during the data transfer. In slave mode, this field must be written as 00h; in master mode, the value of this field is driven on the chip select pins, i.e. bit[(16+n):16] are put out on $\overline{\text{SPISCS[n]}}$ to $\overline{\text{SPISCS[0]}}$ respectively during a transfer. Please see the device-specific data manual for the supported number of the chip select pins. |
| 15-0 | TXDATA | Transfer data (value = 0-FFFFh). When written, these bits will be copied to the shift register if it is empty. If the shift register is not empty, the TXBUF will hold the written values.<br>SPIGCR1.ENABLE must be set to 1 before this register can be written to. Writing a 0 to the SPIGCR1.ENABLE forces the lower 16 bits of the SPIDAT1 to 0.<br>**Note:** Irrespective of the character length, the transmit data should be right-justified before writing to SPIDAT1. |

## 3.9 SPI Receive Buffer Register (SPIBUF)

**Figure 3-9 SPI Buffer Register (SPIBUF)**




LEGEND: R/W = Read/Write; R = Read only; C = Clear; S = Set; -n = value after reset

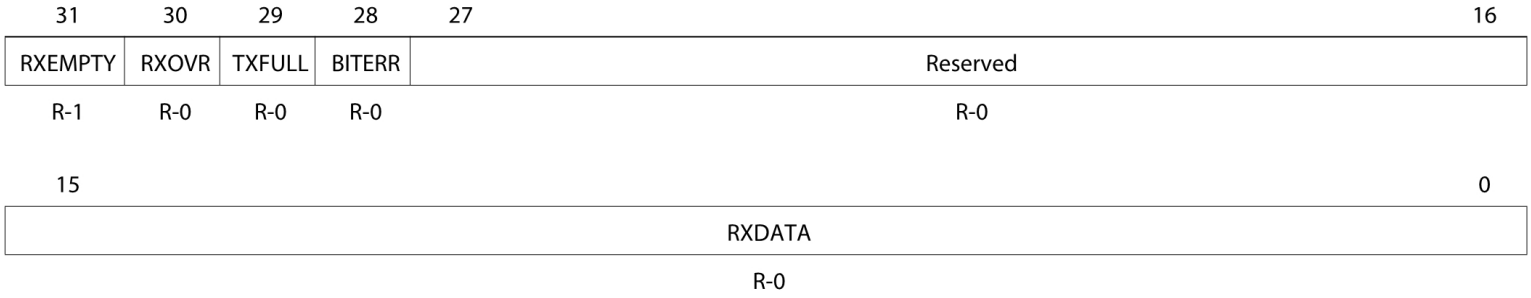
**Table 3-10 SPI Buffer Register (SPIBUF) Field Descriptions**

| Bit | Field | Description |
|---|---|---|
| 31 | RXEMPTY | Receive data buffer empty. When the host reads the SPIBUF field or the whole SPIBUF register, this will automatically set the RXEMPTY flag. When a data transfer is completed, the received data is copied into SPIBUF, the RXEMPTY flag is cleared. This flag is set to 1 under following conditions:<br>• Reading the RXDATA portion of the SPIBUF register.<br>• Writing 1 to clear the RXINTFLG bit in SPIFLG register.<br>0 = New data has been received and copied into the SPIBUF register.<br>1 = No data received since last reading of SPIBUF register.<br>Write-clearing the SPIFLG.RXINTFLG bit before reading the SPIBUF indicates the received data is being ignored. Conversely, SPIFLG.RXINTFLG can be cleared by reading the RXDATA portion of the SPIBUF register or the entire SPIBUF register. |
| 30 | RXOVR | Receive data buffer overrun. When a data transfer is completed and the received data is copied into the RXBUF while it is already full, RXOVR is set. An overrun always occurs to the RXBUF, and SPIBUF contents never get overwritten until after it is read by the CPU/DMA.<br>Reading SPIBUF register does not clear the RXOVR bit. If an overrun interrupt is detected, then the SPIBUF may need to be read twice to get to the overrun buffer. This is due to the fact that the overrun will always occur to the internal RXBUF. Each read to the SPIBUF will result in RXBUF contents (if it is full) getting copied to SPIBUF.<br>**Note:** A special condition under which RXOVR flag gets set. If both SPIBUF and RXBUF are already full and while another buffer receive is underway, if any errors like TIMEOUT, BITERR and DLENERR occur, then RXOVR will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a normal receiver overrun.<br>0 = No receive data overrun condition occurred since last time reading the data field.<br>1 = A receive data overrun condition occurred since last time reading the data field. |
| 29 | TXFULL | Transmit data buffer full. This flag is a read-only flag. Writing into SPIDAT0 or SPIDAT1 field while the TX shift register is full will automatically set the TXFULL flag. Once the data is copied to the shift register, the TXFULL flag will be cleared. Writing to the SPIDAT0/SPIDAT1 register when both TXBUF and the TX shift register are empty does not set the TXFULL flag.<br>0 = The transmit buffer is empty; SPIDAT0/SPIDAT1 is ready to accept a new data.<br>1 = The transmit buffer is full; SPIDAT0/SPIDAT1 is not ready to accept a new data. |
| 28 | BITERR | Bit error. There was a mismatch of internal transmit data and transmitted data. The SPI samples the signal of the transmit pin (master: SIMO, slave: SOMI) at the receive point (half clock cycle after transmit point). If the sampled value differs from the transmitted value, a bit error is detected and the flag BITERR is set. A possible reason for a bit error can be noise, a too-high bit rate/capacitive load, or another master/slave trying to transmit at the same time.<br>**Note:** This flag is cleared to 0 when RXDATA portion of the SPIBUF register is read.<br>0 = No bit error occurred.<br>1 = A bit error occurred. |
| 27-16 | Reserved | Reads return 0 and writes have no effect. |
| 15-0 | RXDATA | SPI receive data (value = 0-FFFFh). This is the received data, transferred from the receive shift-register at the end of a transfer completion. Irrespective of the programmed character length and the direction of shifting, the received data is stored right-justified in the register. |

## 3.10 SPI Emulation Register (SPIEMU)

**Note—**All the fields of SPIEMU register are Read-Only. Read operation on this register under any mode will not have any impact on the status of this or any other registers.

**Figure 3-10 SPI Emulation Register (SPIEMU)**




Legend: R = Read only; -*n* = value after reset

**Table 3-11 SPI Emulation Register (SPIEMU) Field Descriptions**

| Bit | Field | Description |
|---|---|---|
| 31 | RXEMPTY | Receive data buffer empty.<br>0 = New data has been received and copied into the SPIBUF register.<br>1 = No data has been received since the last time the SPIBUF register was read. |
| 30 | RXOVR | Receive data buffer overrun.<br>0 = No receive data overrun condition occurred since last time the data field was read.<br>1 = A receive data overrun condition occurred since the last time the data field was read. |
| 29 | TXFULL | Transmit data buffer full.<br>0 = The transmit buffer is empty; SPIDAT0/SPIDAT1 is ready to accept new data.<br>1 = The transmit buffer is full; SPIDAT0/SPIDAT1 is not ready to accept new data. |
| 28 | BITERR | Mismatch of internal transmit data and transmitted data.<br>0 = No bit error occurred.<br>1 = A bit error occurred. |
| 27-16 | Reserved | Reads return 0 and writes have no effect. |
| 15-0 | RXDATA | SPI receive data (value = 0-FFFFh). SPI emulation is a mirror of the SPIBUF register. The only difference between SPIEMU and SPIBUF is that a read from SPIEMU does not clear any of the status flags. |

## 3.11 SPI Delay Register (SPIDELAY)

**Figure 3-11 SPI Delay Register (SPIDELAY)**

| 31 – 24 | 23 – 16 |
|---|---|
| C2TDELAY | T2CDELAY |
| R/W-0 | R/W-0 |

| 15 – 0 |
|---|
| Reserved |
| R-0 |

Legend: R = Read only; W = Write only; -*n* = value after reset

**Table 3-12 SPI Delay Register (SPIDELAY) Field Descriptions**

| Bit | Field | Description |
|---|---|---|
| 31-24 | C2TDELAY | Chip-select-active-to-transmit-start delay (value = 0-FFh). C2TDELAY is used in master mode only. It defines a setup time for the slave device that delays the data transmission from the chip select active edge by a multiple of SPI module clock cycles. C2TDELAY can be configured between 2 and 257 SPI module clock cycles. See Figure 3-12.<br>The setup time value is calculated as follows:<br>$t_{C2TDELAY}$ = (C2TDELAY + 2) × SPI module clock period<br>**Note:** If C2TDELAY = 0, then $t_{C2TDELAY}$ = 0.<br>***Example:*** SPI module clock = 25 MHz -> SPI module clock period = 40 ns; C2TDELAY = 06h; then $t_{C2TDELAY}$ = 320 ns;<br>When the chip select signal becomes active, the slave has to prepare for data transfer within 320 ns.<br>**Note:** If phase = 1, the delay between $\overline{\text{SPISCS[n]}}$ falling edge to the first edge of $\overline{\text{SPISCS[n]}}$ will have an additional 0.5 SPICLK period delay. This delay is as per the SPI protocol. |
| 23-16 | T2CDELAY | Transmit-end-to-chip-select-inactive delay (value = 0-FFh). T2CDELAY is used in master mode only. It defines a hold time for the slave device that delays the chip select deactivation by a multiple of SPI module clock cycles after the last bit is transferred. T2CDELAY can be configured between 2 and 256 SPI module clock cycles.<br>See Figure 3-13.<br>The hold time value is calculated as follows:<br>$t_{T2CDELAY}$ = (T2CDELAY + 1) × SPI module clock period<br>Note: If T2CDELAY = 0, then $t_{T2CDELAY}$ = 0<br>***Example:*** VBUSPCLK = 25 MHz -> VBUSPCLK period = 40 ns; T2CDELAY = 03h; then $t_{T2CDELAY}$ = 160 ns;<br>After the last data bit (or parity bit) is being transferred the chip select signal is held active for 160 ns.<br>**Note:** If phase = 0, then between the last edge of SPICLK and rise-edge of $\overline{\text{SPISCS[n]}}$ there will be an additional delay of 0.5 SPICLK period. This is as per the SPI protocol. |
| 15-0 | Reserved | Reads return 0 and writes have no effect. |

**Figure 3-12 Example: $t_{C2TDELAY}$ = 8 SPI Module Clock Cycles**

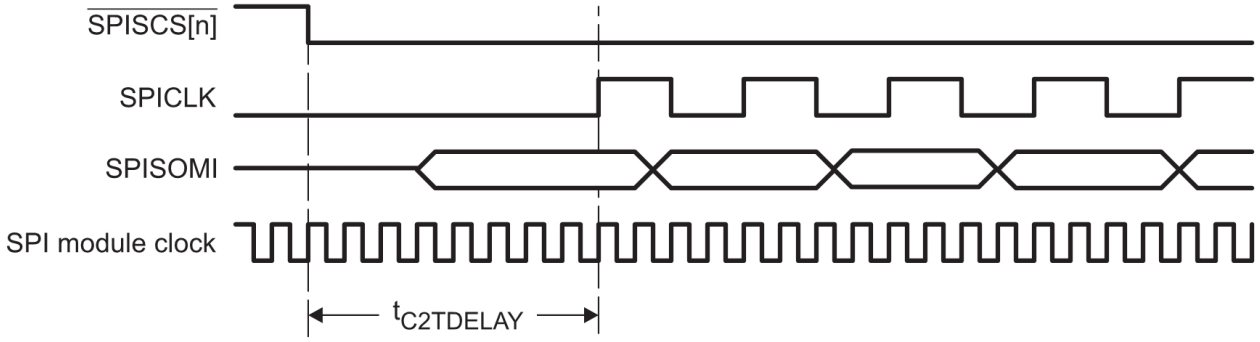
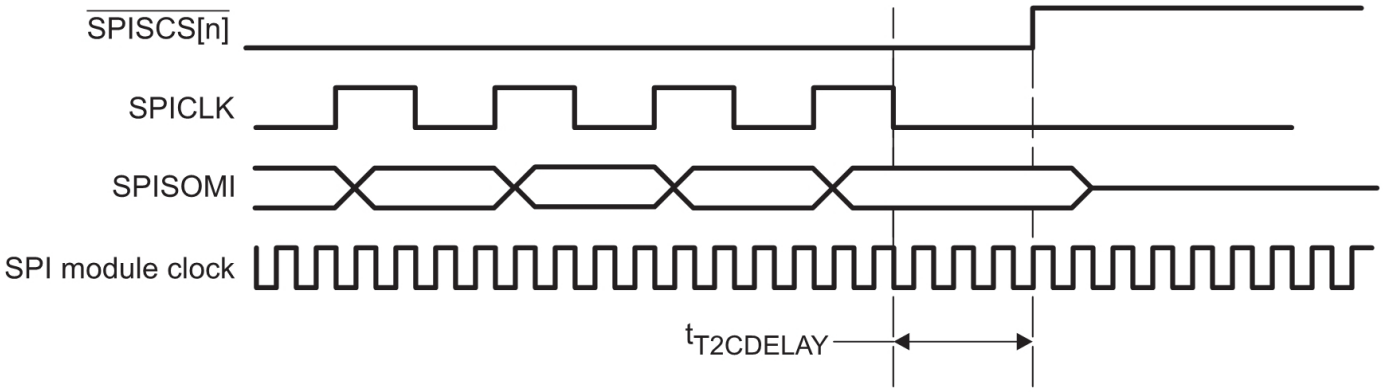
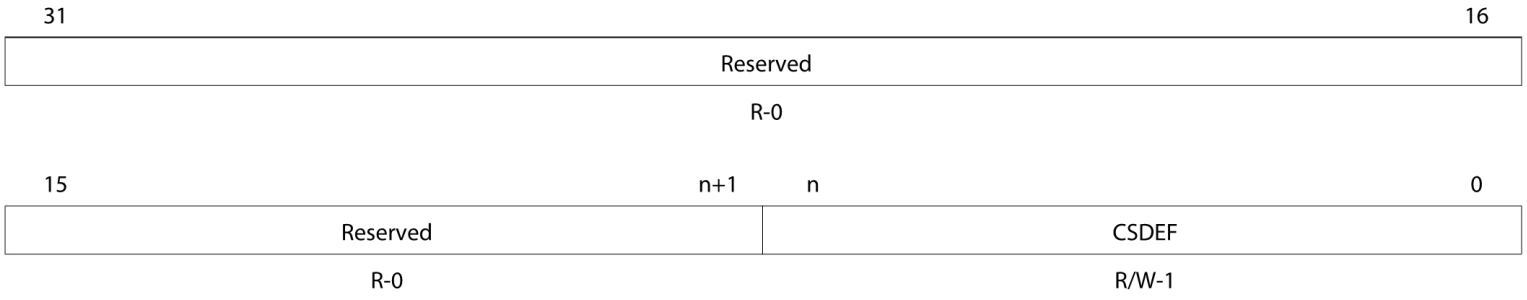


**Figure 3-13 Example: $t_{T2CDELAY}$ = 4 SPI Module Clock Cycles**




## 3.12 SPI Default Chip Select Register (SPIDEF)

**Figure 3-14 SPI Default Chip Select Register (SPIDEF)**




Legend: R = Read only; W = Write only; -*n* = value after reset

**Table 3-13 SPI Default Chip Select Register (SPIDEF) Field Descriptions**

| Bit | Field | Description |
|---|---|---|
| 31-(n+1) | Reserved | Reads return 0 and writes have no effect. |
| n-0 | CSDEF | Chip Select Default pattern. In master mode, the bit position *i* in this field determines the $\overline{\text{SPISCS[i]}}$ pin state when no transmissions are currently performed. It allows the user to set a chip select pattern which deselects all the SPI slaves. Please see the device-specific data manual for supported number of the chip select pins.<br>Bit position *i* in this field:<br>0 = The corresponding $\overline{\text{SPISCS[i]}}$ pin is cleared to 0 when no transfer occurs.<br>1 = The corresponding $\overline{\text{SPISCS[i]}}$ pin is set to 1 when no transfer occurs. |

## 3.13 SPI Data Format Registers (SPIFMT*n*)

**Figure 3-15 SPI Data Format Register (SPIFMT*n*)**

| 31 30 | 29 24 | 23 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| Reserved | WDELAY | Reserved | SHIFTDIR | Reserved | DISCSTIMERS | POLARITY | PHASE |
| R-0 | R/W-0 | R-0 | R/W-0 | R-0 | R/W-0 | R/W-0 | R/W-0 |

| 15 8 | 7 5 | 4 0 |
|---|---|---|
| PRESCALE | Reserved | CHARLEN |
| R/W-0 | R-0 | R/W-0 |

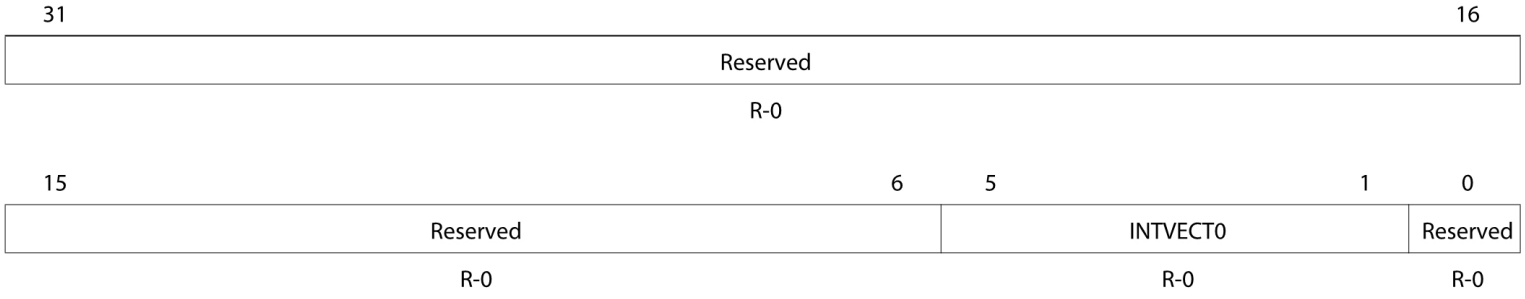
Legend: R = Read only; W = Write only; -*n* = value after reset

**Table 3-14 SPI Data Format Register (SPIFMT*n*) Field Descriptions**

| Bit | Field | Description |
|---|---|---|
| 31-30 | Reserved | Reads return 0 and writes have no effect. |
| 29-24 | WDELAY | Delay in between transmissions. Idle time that will be applied at the end of the current transmission if the bit WDEL is set in the current buffer. The delay to be applied is equal to:<br>$WDELAY \times P_{SPI\ module\ clock} + 2 \times P_{SPI\ module\ clock}$<br>$P_{SPI\ module\ clock}$ -> Period of SPI module clock |
| 23-21 | Reserved | Reads return 0 and writes have no effect. |
| 20 | SHIFTDIR | Shift direction.<br>0 = Most significant bit is shifted out first.<br>1 = Least significant bit is shifted out first. |
| 19 | Reserved | Reads return 0 and writes have no effect. |
| 18 | DISCSTIMERS | Disable chip select timers for this format register. The C2TDELAY and T2CDELAY timers are by default enabled for all the data format registers. Using this bit, these timers can be disabled for a particular data format if not required. When a master is handling multiple slaves, with varied set-up hold requirement, the application can selectively choose to include or not include the chip select delay timers for any slaves.<br>0 = Both C2TDELAY and T2CDELAY counts are inserted for the chip selects.<br>1 = No C2TDELAY or T2CDELAY is inserted in the chip select timings. |
| 17 | POLARITY | SPI clock polarity.<br>0 = SPI clock signal is low-inactive (before and after data transfer the clock signal is low).<br>1 = SPI clock signal is high-inactive (before and after data transfer the clock signal is high). |
| 16 | PHASE | SPI clock delay.<br>0 = SPI clock signal is not delayed versus the transmit/receive data stream. The first data bit is transmitted with the first clock edge and the first bit is received with the second (inverse) clock edge.<br>1 = SPI clock signal is delayed by a half SPI clock cycle versus the transmit/receive data stream. The first transmit bit has to output prior to the first clock edge. The master and slave receive the first bit with the first edge. |
| 15-8 | PRESCALE | SPI prescaler (value = 0-FFh). It determines the bit transfer rate if the SPI is the network master and is directly derived from the SPI module clock. If the SPI is configured as slave, PRESCALE does not need to be configured.<br>The clock rate can be calculated as:<br>SPI clock frequency = SPI module clock/(PRESCALE + 1)<br>**Note:** When PRESCALEx is cleared to 0, the SPI clock rate defaults to SPI module clock/2. |
| 7-5 | Reserved | Reads return 0 and writes have no effect. |
| 4-0 | CHARLEN | SPI data word length (value = 0-1Fh). Legal values are 2h (data word length = 2 bit) to 10h (data word length = 16). Illegal values, such as 0 or 1Fh are not detected and their effect is indeterminate. |

## 3.14 SPI Interrupt Vector Register 0 (INTVEC0)

**Figure 3-16 SPI Interrupt Vector Register 0 (INTVEC0)**



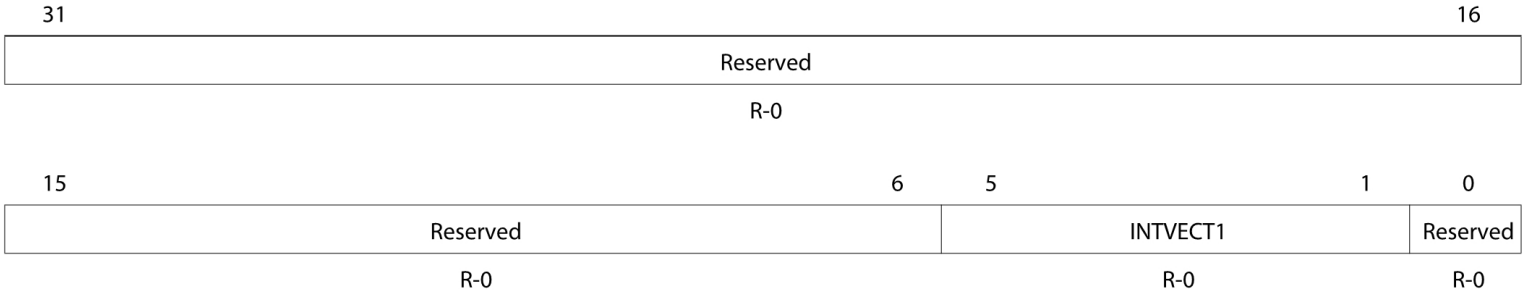

Legend: R = Read only; -*n* = value after reset

**Table 3-15 SPI Interrupt Vector Register 0 (INTVEC0) Field Descriptions**

| Bit | Field | Description |
|---|---|---|
| 31-6 | Reserved | Reads return 0 and writes have no effect. |
| 5-1 | INTVECT0 | Interrupt vector for interrupt line INT0. INTVECT0 returns the vector of the pending interrupt at interrupt line INT0. If more than one interrupt is pending, INTVECT0 always references the highest priority interrupt source first. The interrupts available for SPI in the descending order of their priorities are as given below.<br>• Transmission error Interrupt<br>• Receive buffer overrun interrupt<br>• Receive buffer full interrupt<br>• Transmit buffer empty interrupt<br>The INTVECT0 field just reflects the status of SPIFLG in a vectorized format. So, any updates to SPIFLG will automatically reflect in the vector value in this register.<br>Vectors for each of these interrupts will be reflected on the INTVECT0 bits, when they occur. Reading the vectors for the receive buffer overrun and receive buffer full interrupts will automatically clear the respective flags in the SPIFLG. Reading the vector register when transmitter empty is indicated does not clear the TXINTFLG in SPIFLG. Writing a new data to SPIDAT0/SPIDAT1 clears the transmitter empty interrupt. On reading the INTVECT0 bits, the vector of the next highest priority interrupt (if any) will be then reflected on the INTVECT0 bits. If two or more interrupts occur simultaneously, the vector for the highest priority interrupt will be reflected on the INTVECT0 bits.<br>The following are the SPI interrupt vectors for line INT0:<br>0 = No interrupt pending<br>1h-10h = Reserved<br>11h = Error interrupt pending. See the lower halfword of SPIINT0 to determine more details about the type of error.<br>12h = The pending interrupt is receive buffer full interrupt.<br>13h = The pending interrupt is receive buffer overrun interrupt.<br>14h = The pending interrupt is transmit buffer empty interrupt.<br>15h-1Fh = Reserved |
| 0 | Reserved | Reads return 0 and writes have no effect. |

## 3.15 SPI Interrupt Vector Register 1 (INTVEC1)

**Figure 3-17 SPI Interrupt Vector Register 1 (INTVEC1)**




Legend: R = Read only; -*n* = value after reset

**Table 3-16 SPI Interrupt Vector Register 1 (INTVEC1) Field Descriptions**

| Bit | Field | Description |
|---|---|---|
| 31-6 | Reserved | Reads return 0 and writes have no effect. |
| 5-1 | INTVECT1 | Interrupt vector for interrupt line INT1. INTVECT1 returns the vector of the pending interrupt at interrupt line INT1. If more than one interrupt is pending, INTVECT1 always references the highest priority interrupt source first. The interrupts available for SPI in the descending order of their priorities are as given below.<br>• Transmission error Interrupt<br>• Receive buffer overrun interrupt<br>• Receive buffer full interrupt<br>• Transmit buffer empty interrupt<br>The INTVECT1 field just reflects the status of SPIFLG in a vectorized format. So, any updates to SPIFLG will automatically reflect in the vector value in this register. Vectors for each of these interrupts will be reflected on the INTVECT0 bits, when they occur. Reading the vectors for the receive buffer overrun and receive buffer full interrupts will automatically clear the respective flags in the SPIFLG. Reading the vector register when transmitter empty is indicated does not clear the TXINTFLG in SPIFLG. Writing a new data to SPIDAT0/SPIDAT1 clears the transmitter empty interrupt. On reading the INTVECT1 bits, the vector of the next highest priority interrupt (if any) will be then reflected on the INTVECT1 bits. If two or more interrupts occur simultaneously, the vector for the highest priority interrupt will be reflected on the INTVECT1 bits.<br>The following are the SPI interrupt vectors for line INT1:<br>0 = No interrupt pending<br>1h-10h = Reserved<br>11h = Error interrupt pending. See the lower halfword of SPIINT0 to determine more details about the type of error.<br>12h = The pending interrupt is receive buffer full interrupt.<br>13h = The pending interrupt is receive buffer overrun interrupt.<br>14h = The pending interrupt is transmit buffer empty interrupt.<br>15h-1Fh = Reserved |
| 0 | Reserved | Reads return 0 and writes have no effect. |

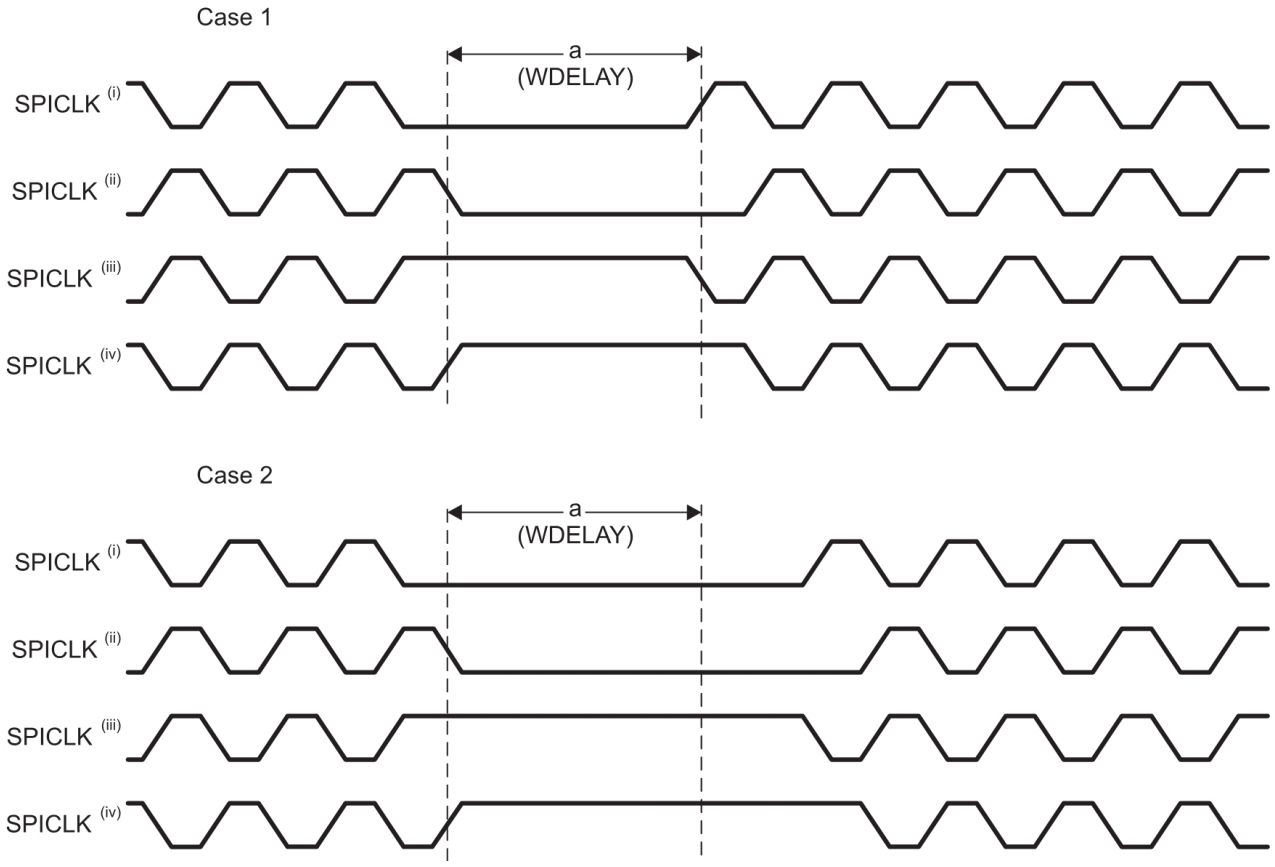
# Timing Diagrams

This appendix contains timing diagrams illustrating the C2TDELAY, T2CDELAY, and WDELAY delays and their interaction with the $\overline{\text{SPISCS[n]}}$ pins for all SPI modes.

## A.1 SPI 3-Pin Mode

Figure A-1 illustrates the WDELAY option in SPI 3-pin master mode. This is the only delay available in this mode. In CASE1, a new transfer is initiated during the WDELAY period and the transfer begins immediately after the WDELAY period ends. In CASE2, while WDELAY has completed, a new transfer will not begin until SPIDAT0/SPIDAT1 have been written with new data.

**Figure A-1 SPI 3-Pin Master Mode with WDELAY**



## A.2 SPI 4-Pin with Chip Select Mode

Figure A-2 illustrates the T2CDELAY, WDELAY and C2TDELAY delays in SPI 4-pin with $\overline{\text{SPISCS[n]}}$ master mode. All the three delay periods T2CDELAY, WDELAY, and C2TDELAY proceed to completion when enabled.

**Figure A-2 SPI 4-Pin with $\overline{\text{SPISCS[n]}}$ Mode with T2CDELAY, WDELAY, and C2TDELAY**